

Pescador:Lógica de RC

Se permite utilizar este documento en los términos estipulados por la licencia [Attribution-NonCommercial-NoDerivs 2.5 Mexico de Creative Commons](#)

Tabla de contenidos

- 1 Situación del documento
- 2 Objetivos de diseño
- 3 Arquitectura general
 - 3.1 Definiciones
 - 3.2 Resumen
- 4 Estado del código
- 5 Estructuras de RC hardcodeadas
- 6 Datos de ejemplo
- 7 Elementos fundamentales
 - 7.1 Descripciones: concepto general
 - 7.2 Conjuntos de reglas
 - 7.3 Descriptores
 - 7.4 Tipos de datos
 - 7.4.1 Los FundamentalDataType
 - 7.4.1.1 LFMarkup
 - 7.4.2 Los ComplexDataType
 - 7.4.3 Los AbstractDataType
 - 7.4.4 Los vectores de comparación
 - 7.4.5 Reglas de herencia en los tipos de datos
 - 7.4.6 Mecanismos de captura
 - 7.5 Relaciones de RC
 - 7.6 Valores de base para la presentación
 - 7.7 Funciones de valores de base para la presentación
 - 7.8 Fragmentos de descripciones
 - 7.9 Tipos y configuración de descripciones
- 8 Operaciones
 - 8.1 Extracción de datos
 - 8.1.1 Descripciones
 - 8.1.1.1 Cualquier descripción menos las descripciones para la modificación de datos
 - 8.1.1.2 Descripciones para la modificación de datos
 - 8.1.2 Generación de conjuntos
 - 8.1.2.1 Conjuntos ordenados de descripciones
 - 8.1.2.2 Conjuntos ordenados de descripciones para operaciones de modificación de datos
 - 8.1.2.3 Conjuntos ordenados de BPV
 - 8.1.2.4 Estructuras de subgrupos
 - 8.1.2.5 Conjuntos de descripciones en subgrupos
 - 8.1.2.6 Conjuntos de descripciones en jerarquía
 - 8.1.2.7 Conjuntos de descripciones en jerarquía para operaciones de modificación de datos
 - 8.1.2.8 Conjuntos de BPV en jerarquía
 - 8.2 Modificación de datos
 - 8.2.1 Generar un objeto descriptible
 - 8.2.2 Borrar un objeto descriptible
 - 8.2.3 Mover un objeto descriptible
 - 8.2.4 Modificar una relación de RC
 - 8.2.5 Modificar simultáneamente múltiples relaciones de RC
 - 8.3 Verificación de estructuras
- 9 Archivos de definiciones

- 9.1 Sintaxis
- 9.2 Resumen
- 9.3 Elementos
 - 9.3.1 abbreviation
 - 9.3.2 AbstractDataType
 - 9.3.3 anchor
 - 9.3.4 Area
 - 9.3.5 bindSOCToRuleSet
 - 9.3.6 bindToRuleSet
 - 9.3.7 Block
 - 9.3.8 Bloques de código Ruby
 - 9.3.9 BPVFunctions
 - 9.3.10 Class
 - 9.3.11 Código encrustado en plantillas de descripciones
 - 9.3.12 comment
 - 9.3.13 comparisonVector
 - 9.3.14 ComplexDataType
 - 9.3.15 CompVectorFunction
 - 9.3.16 CompVectorFunctions
 - 9.3.17 Conditions
 - 9.3.18 copyOfTemplate
 - 9.3.19 CreateTemplate
 - 9.3.20 defaultImageBPVFunction
 - 9.3.21 defaultOrderBy
 - 9.3.22 defaultTextBPVFunction
 - 9.3.23 DefinitionSet
 - 9.3.24 dependsRealm
 - 9.3.25 Descriptions
 - 9.3.26 Descriptor
 - 9.3.27 df
 - 9.3.27.1 df en las plantillas de descripción fija
 - 9.3.27.2 df en las secciones VariableDesc
 - 9.3.28 Documentación incrustada multilingüe
 - 9.3.29 externalOntology
 - 9.3.30 FullDesc
 - 9.3.31 FundamentalDataType
 - 9.3.32 generalArchivalThings
 - 9.3.33 hasGroupDomain
 - 9.3.34 Identificador de objeto
 - 9.3.35 ImageBPVFunction
 - 9.3.36 imageBPV
 - 9.3.37 InferenceRule
 - 9.3.38 InferenceRules
 - 9.3.39 KRRelationRule
 - 9.3.40 KRRelationRules
 - 9.3.41 label
 - 9.3.42 linkTo
 - 9.3.43 LIT
 - 9.3.44 mainLevel
 - 9.3.45 maxCardinality
 - 9.3.46 maxNonInfCardinality
 - 9.3.47 minCardinality
 - 9.3.48 ModifyTemplate
 - 9.3.49 multilingual

- 9.3.50 multipleOrdered
- 9.3.51 NonDefaultDescFragments
- 9.3.52 orderUsing
- 9.3.53 OUT
- 9.3.54 outClass
- 9.3.55 outDataType
- 9.3.56 outFromGroup
- 9.3.57 outRequirement
- 9.3.58 outValue
- 9.3.59 Plantilla de descripción fija
- 9.3.60 Property
- 9.3.61 property
- 9.3.62 range
- 9.3.63 Realm
- 9.3.64 Referencia a un BPV
- 9.3.65 Referencia a un RDFNode
- 9.3.66 Referencia a una KRRelationRule
 - 9.3.66.1 Sintaxis básico
 - 9.3.66.2 Encadenamiento de reglas
 - 9.3.66.3 Inversión de reglas
 - 9.3.66.4 Relaciones no inferenciadas
 - 9.3.66.5 Uso en bloques de código
 - 9.3.66.6 Generación de estructuras desde el código
- 9.3.67 Reglas de estructura
- 9.3.68 RepositoryArea
- 9.3.69 RepositorySetup
- 9.3.70 ROOT
- 9.3.71 RuleSet
- 9.3.72 secondaryLevel
- 9.3.73 sharedArchivalThings
- 9.3.74 ShortDesc
- 9.3.75 SOC
- 9.3.76 Structure
- 9.3.77 StructRules
- 9.3.78 Solicitudes de relaciones de RC concretas
- 9.3.79 subClassOf
- 9.3.80 subDataTypeOf
- 9.3.81 subPropertyOf
- 9.3.82 systemThings
- 9.3.83 TextBPVFunction
- 9.3.84 textBPV
- 9.3.85 Texto multilingüe
- 9.3.86 thisOntology
- 9.3.87 Title
- 9.3.88 URI de los recursos RDF
- 9.3.89 UniqueDF
 - 9.3.89.1 UniqueDF en las plantillas de descripción fija
 - 9.3.89.2 UniqueDF en en las secciones VariableDesc
- 9.3.90 uniqueImageBPV
- 9.3.91 uniqueTextBPV
- 9.3.92 useDefinition
- 9.3.93 VariableDesc
- 9.3.94 Vocabulary
- 10 Notas

Este documento describe las características principales del sistema de lógica de repositorio de conocimiento (RC) que se desarrolla para el Pescador.

Situación del documento

El documento constituye una propuesta de diseño inicial. Conforme avance el desarrollo del sistema Pescador, las adecuaciones que se tengan que hacer al diseño se integrarán al texto, de modo que al final éste quede como documentación para programadores y administradores de sitios que emplean el sistema.

Objetivos de diseño

Este sistema de lógica de RC establece: (1) operaciones y reglas generales para un sistema de difusión, investigación y gestión de fondos digitales; y (2) mecanismos de configuración que permiten fijar los detalles de la aplicación de dichas operaciones y reglas con fondos específicos. Con este diseño se busca:

- Cuando sea apropiado, reutilizar operaciones, reglas y configuraciones en distintas partes de la lógica de RC
- Permitir la implementación de una amplia variedad de funciones para el usuario
- Permitir que dichas funciones sean simultáneamente complejas, flexibles y de uso intuitivo
- Permitir un *back-end* de representación de conocimiento compleja y expresiva
- Hasta sea apropiado, reunir en un sólo lugar (o en pocos lugares) la información de configuración
- Ofrecer mecanismos de configuración sencillos, intuitivos, no repetitivos y modulares
- Establecer los mecanismos de configuración más apropiados para cada parte de la lógica de RC; en algunos casos, configurar las operaciones y reglas mediante estructuras semánticamente ricas y traducibles a formatos estándares; en otros, permitir la incrustación de código que implemente operaciones complejas, cuya representación con estructuras semánticamente ricas sería engorrosa
- Permitir la optimización de las operaciones para repositorios grandes
- Permitir la integración de la lógica de RC a una arquitectura de aplicación apropiada

Arquitectura general

Definiciones

Repositorio de conocimiento (RC)

Conjunto de información estructurado según las normas de la Web Semántica. El Pescador guarda dicha información en un almacén de triples.

Formato de datos

Un modo de organización de datos.

Formato de presentación al usuario (FPU)

Un formato de datos que permite la presentación de éstos directamente ante el usuario.

FPU interactivo estándar

El FPU estándar que despliega el interfaz Web del Pescador. Incluye elementos interactivos como vínculos, botones, pestañas y demás controles.

FPU para edición

El FPU que se emplea en el interfaz que permite al usuario modificar los datos del RC.

FPU no interactivo estándar

El FPU estándar para soportes visuales no interactivos, como el papel o un archivo de procesador de palabras.

Formato intermedio

Un formato de datos que no es un FPU, sino el resultado de un paso intermedio en una cadena de transformaciones de datos en el sistema.

Lógica de RC

Operaciones y reglas de la manipulación del RC y de los datos que de él provienen, y mecanismos de configuración de dichas operaciones y reglas.

Capa de lógica de RC

Capa del Pescador que implementa la lógica de RC.

Objeto descriptible de RC

Recurso definido en el RC que es susceptible de ser descrito con base en los datos contenidos en el mismo RC.

Archivo de definiciones

El mecanismo de configuración básico de la lógica de RC; configura vocabulario, reglas y operaciones de la misma.

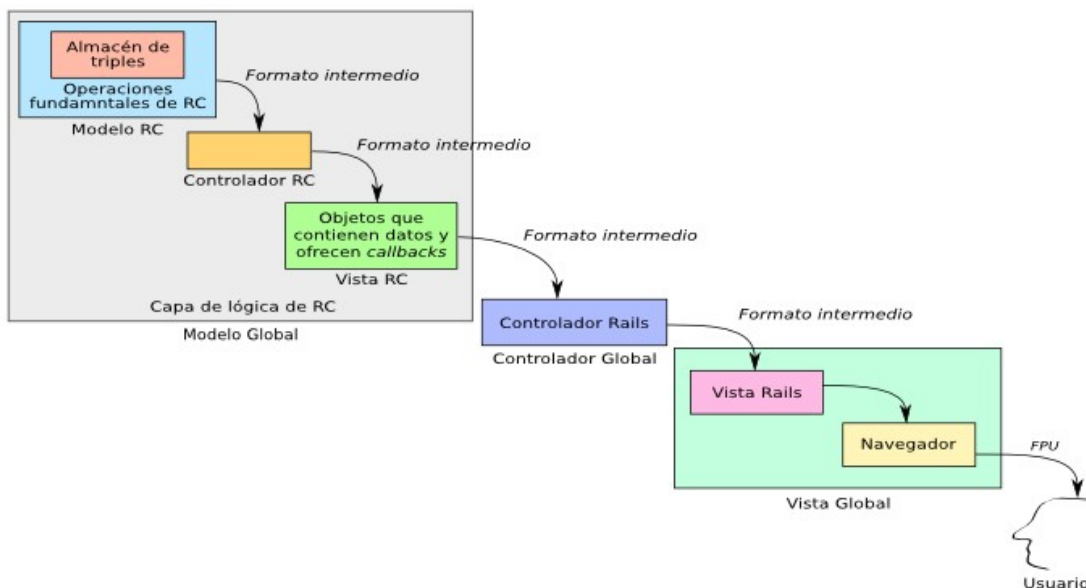
Resumen

La arquitectura del sistema sigue el patrón modelo-vista-controlador encrustado (*nested model-view-controller*). En el nivel global existe un modelo, una vista y un controlador. En este esquema la capa de lógica de RC representa el modelo global, y al interior de ésta se repite el mismo patrón modelo-vista-controlador.

La información acerca de los fondos se registra en el almacén de triples, que es el RC propiamente dicho y forma parte del modelo al interior de la capa de lógica de RC.

Esto se muestra en la Figura 1. En el diagrama las flechas indican el camino que siguen datos para desplazarse del RC hasta el usuario, y el texto en cursivas señala el formato de los datos en ese proceso de transferencia.

Figura 1. Capas lógicas del sistema



Los cuatro tipos de operación que lleva a cabo el sistema con los datos del RC son:

1. Extraer datos del RC, reorganizarlos en un FPU u otro formato, y enviarlos al usuario u otro programa.
2. Modificar los contenidos del RC con base en las indicaciones del usuario.
3. Asegurar la coherencia de las estructuras del RC.
4. Configurar, con base en las indicaciones del usuario, la capa de lógica de RC y la estructura del mismo.

Cada uno de estos tipos generales engloba diversas categorías de operación más específicas, las cuales se describen en el apartado [Operaciones](#). Nótese que únicamente se han proyectado los detalles del funcionamiento de los primeros tres tipos, dejando para una etapa posterior el diseño del último.

Estado del código

El estado actual del código de la capa de lógica de RC es: se ha elaborado una propuesta de jerarquía de clases así como una descripción breve de cada clase, y se está trabajando en la implementación de las funciones básicas.

Puede consultarse en línea tanto la [documentación Javadoc](#) como las [gráficas de jerarquía de clases](#). En las explicaciones que siguen a menudo se incluyen vínculos a la documentación de clases determinadas.

El depósito de código svn puede consultarse [aquí](#) y [aquí](#).

Estructuras de RC hardcodeadas

El sistema *no* fue diseñado para gestionar cualquier representación de conocimiento, sino una que se estructure según una serie de reglas establecidas por el mismo sistema.

El primer requisito de estructura se refiere a la organización general del RC, el cual se compone de varias ontologías interrelacionadas. Casi todas contienen definiciones de términos y reglas, es decir, son "esquemas" (*schema*), mientras una de las ontologías, **repository**, almacena la mayoría de los objetos descriptibles de RC así como los datos que los describen. Esto significa que la mayoría de los [individuos](#) (según la terminología de OWL) se ubican en la ontología **repository**.

Internamente **repository** se divide en "áreas de repositorio"; todos los recursos descriptibles de RC pertenecen a una área de repositorio u otra.

El funcionamiento de la lógica de RC se configura por medio de archivos de definiciones (con extensión `.def`). Cada archivo de definiciones genera automáticamente una ontología tipo esquema que se incluye en el RC, y al mismo tiempo genera una área de repositorio al interior de **repository**, donde se colocan los individuos creados por ese archivo de definiciones.

Existen dos archivos de definiciones que juegan un papel fundamental en el funcionamiento del sistema: **sistem.def** y **shared-worldview.def**. Siempre deben incluirse ambos archivos en el conjunto de archivos de definiciones que emplea el sistema.¹ En tanto, forman parte del RC las ontologías generadas por estos archivos, y se crean en la ontología **repository** las áreas de repositorio correspondientes. La función principal de **sistem.def** es establecer los parámetros del almacenamiento y descripción, en el RC, de objetos propios del sistema (como usuarios y permisos).

Y `shared-worldview.def` establece los parámetros del almacenamiento y descripción de datos que pueden compartirse entre diversos fondos digitales del RC (por ejemplo, lugares geográficos o tesauros estandarizados). Asimismo `shared-worldview.def` proporciona términos y conceptos comunes (como la noción de título y referencias al tiempo) que pueden retomarse en la descripción de diversos fondos.

Además de los archivos `system.def` y `shared-worldview.def` típicamente existen archivos de definiciones adicionales, uno por cada entidad (persona o institución) responsable de una o más colecciones en el sistema. Estos archivos definen los mecanismos específicos de cada fondo, y a partir de ellas también se generan ontologías y áreas de repositorio. La Figura 2 presenta este esquema de manera visual.

Figura 2. Estructura de ontologías



Respecto a la organización interna que se ha definido para `repository`, las divisiones internas que ya mencionamos, las áreas de repositorio, de hecho forman únicamente el nivel más general de la estructura de esa ontología. Al interior de cada área existen objetos denominados "colecciones para la organización del sitio". Todos los objetos descriptibles forman parte de una (y sólo una) colección para la organización del sitio. Los objetos descriptibles siempre pertenecen al mismo área de repositorio que la colección de la cual forman parte.

El sistema permite asignar permisos (o prohibiciones) de lectura y escritura distintas para cada área de repositorio. Así se determina cuáles usuarios tienen el derecho de ver o modificar los contenidos de cualquiera de ellas.

La ontología `repository` siempre contiene tres o más áreas: (1) el área generada por el archivo `system.def`, donde se guardan los datos internos del sistema ; (2) el área generada por el archivo `shared-worldview.def`, donde se almacenan datos compartidos entre todos los fondos digitales del RC; y (3) una área por cada fondo o conjunto de fondos bajo la responsabilidad de una misma entidad, generadas por los archivos de definiciones correspondientes.

Figura 3. Estructura de `repository`

La estructura de las subgráficas que describen los objetos descriptibles se establece por medio de los archivos de definición, cuyo funcionamiento se explica en los siguientes apartados de este documento.

Datos de ejemplo

Para explicar las operaciones, reglas y mecanismos de configuración del sistema de lógica de RC, se emplearán los datos de un fondo fotográfico imaginario, registrados en los siguientes archivos:

[repository.n3](#)

Los contenidos de la ontología `repository` para un RC que almacena el fondo, en formato [n3](#). (Acerca del problema de los literales que tienen simultáneamente un tipo de datos y un idioma, véase la sección acerca de los [descriptores](#) y la [nota 2](#).)

[archivo-juan-perez.def](#)

Un archivo de definiciones específico para este fondo.

[shared-worldview.def](#)

Una versión reducida del archivo de definiciones para datos y reglas compartidas. Ésta no es una versión definitiva del archivo con ese nombre que forma parte del sistema.

En los datos de ejemplo se hace referencia a vocablos y reglas que se definen en `system.def`, archivo que no se ha incluido aquí ya que aún no se ha elaborado.

Elementos fundamentales

A continuación se describen los elementos fundamentales que se emplean en las operaciones de la lógica de RC.

Descripciones: concepto general

Las descripciones ([Description](#)) son objetos que describen un objeto descriptible de RC. Son un componente fundamental de otros objetos que puede enviar la capa de lógica de RC al controlador global en el proceso de extracción de datos del RC.

Las descripciones que produce la capa de lógica de RC emplean un formato de datos intermedio denominando *formato de base para la presentación (base presentation format)*. Este formato de datos permite plasmar todos los componentes de la descripción de una manera genérica. Después de recibir la descripción en este formato, el controlador global la envía a la vista global, que la transforma de en un formato apropiado para un medio determinado (por ejemplo, pantalla o papel). Es decir, a partir de las descripciones genéricas que genera la capa de lógica de RC se puede crear el FPU interactivo estándar o el FPU no interactivo estándar (u otros tipos de FPU).

Los componentes de las descripciones incluyen los [valores de base para la presentación](#) y diversos tipos de elementos estructurales, como áreas, bloques, etcétera. (Véase el apartado [Tipos y configuración de descripciones](#) para una explicación más detallada.)

Conjuntos de reglas

Los conjuntos de reglas ([RuleSet](#)) permiten agrupar las configuraciones de operaciones de lógica de RC. Normalmente un conjunto de reglas constituye un todo coherente que puede vincularse con diversos objetos en el RC y así determinar los mecanismos de procesamiento de esos objetos. Pueden vincularse con:

- los valores de un determinado tipo de datos;
- todos los miembros de una colección para la organización del sitio determinada; o
- individuos específicos.

(En una etapa posterior también se permitirá vincular los **RuleSet** con todos los individuos de una clase determinada al interior de una colección para la organización del sitio determinada.)

Presentamos a continuación un ejemplo breve con la finalidad de aclarar el concepto general. Los detalles del funcionamiento de los **RuleSet** y las directrices que contienen se explican más adelante en el presente texto.

En el siguiente fragmento del archivo [shared-worldview.def](#) se define un **RuleSet** denominado **person**.

```
<RuleSet person>
  <Structure>
    <Descriptor has_given_names>
      property swv:hasGivenNames
      minCardinality 1
      maxCardinality 1
      outRequirement fromPropRange
    </Descriptor>
    <Descriptor has_last_names>
      property swv:hasLastNames
      minCardinality 1
```



```
taxis"@en^xsd:string ] .
```

Aquí consideramos que existe un descriptor con el predicado `swv:hasName`. De hecho, el descriptor no sólo mapea a más de un triple--además, corresponde a una gráfica RDF diferente según el idioma que se usa. Si el idioma es inglés, el descriptor mapea a la ruta de triples que va desde el recurso `rep:archival_thing_10` hasta el literal "Bicycle taxis"; pero si el idioma es español, mapea a los triples que van desde `rep:archival_thing_10` hasta "Bicitaxis".

En el mismo archivo se presenta un ejemplo de cómo se establece un orden predeterminado para los descriptors que emplean el mismo sujeto y el mismo predicado:

```
rep:archival_thing_22    a swv:Image ;  
                        (...)   
                        swv:hasTopic [          a sys:MultOrdObjectsSeq;  
                                                rdf:_1 rep:archival_thing_8;  
                                                rdf:_2 rep:archival_thing_9;  
                                                rdf:_3 rep:archival_thing_10 ] .
```

Aquí hay tres descriptors con el sujeto `rep:archival_thing_22` y el predicado `swv:hasTopic`, y estos descriptors tienen un orden establecido, de modo que el descriptor que tiene como objeto `rep:archival_thing_8` es el primero, después viene el que remite a `rep:archival_thing_9`, etcétera. Pero como RDF en sí no admite el ordenamiento de los triples que partan de un recurso determinado, el RC debe emplear un contenedor RDF para registrar el orden; el resultado de ello es que cada descriptor corresponde a más de un triple.

En los archivos de definiciones se establece cuáles descriptors pueden o deben vincularse con cuáles objetos descriptibles, y cuáles opciones (entre ellas, las que se acaban de exponer) se encuentran disponibles en cada caso.

Es importante notar que los descriptors no representan modificación alguna a las normas de la Web Semántica, sino que sólo constituyen un atajo que permite establecer y atravesar sencillamente estructuras comunes. Por lo tanto, todas las funciones que ofrecen deben tener como estructura de fondo una gráfica RDF que represente dichas funciones según la "mejor práctica" (*best practice*) de diseño de ontologías en WS.

Tipos de datos

Las especificaciones de la Web Semántica establecen mecanismos para asignar un tipo de datos a un nodo literal. La lógica de RC del Pescador amplía este concepto: permite la definición de tipos de datos complejos cuyos valores son un conjunto de triples estructurados como árbol, con un nodo raíz. En la lógica de RC estas dos categorías generales de tipos de datos se llaman `FundamentalDataType` y `ComplexDataType` respectivamente. Una tercera categoría de tipos de datos es `AbstractDataType`, cuyos miembros no pueden ser asignados directamente a elementos del RC, sino que funcionan como ancestros de otros tipos de datos en una jerarquía de sub/supertipos de datos. Los archivos de definiciones permiten establecer tipos de datos que pertenezcan a cualquiera de estas tres categorías.

Los FundamentalDataType

En el RC, para asignar un `FundamentalDataType` a un literal se utilizan los mecanismos definidos por las especificaciones de la WS.²

En la ontología `repository`, tal como muestra [repository.n3](#), todos los literales que constituyen el objeto de un descriptor deben tener un tipo de datos de la categoría

FundamentalDataType.

En los archivos de definiciones se pueden establecer reglas de validación de datos para capturar valores del tipo `FundamentalDataType`.

Uno de los tipos de datos de esta categoría que se define en el archivo `system.def` es `LFMarkup`, el cual se describe a continuación.

LFMarkup

El nombre de este tipo de datos significa "marcado para vincular y formatear" ("*Linking and Formatting Markup*"). Puede considerarse un descendiente de `rdf:XMLLiteral`, y sus valores deben cumplir con los parámetros establecidos para éste. (Esto significa que sus valores del espacio léxico siempre constituyen un fragmento de XML "bien equilibrado".³) A continuación se describen las etiquetas que puede contener:

`krpointer`

Un vínculo a un objeto describable de RC. Acepta los siguientes atributos:

`uri`

El uri completo (sin abreviaciones de espacio de nombres) del objeto describable.

`autotext`

Un atributo booleano. Indica que al presentar el vínculo al usuario el texto del mismo será el resultado de la [función de texto VBP](#) por default (véase la explicación más adelante) del recurso señalado.

Si la etiqueta `kr` no tiene un atributo `autotext` entonces debe aparecer como elemento no vacío, cuyos contenidos se presentarán al usuario como texto del vínculo.

Por ejemplo, la siguiente etiqueta desplegará ante el usuario un vínculo al recurso señalado, con el resultado de la función de texto VBP por default como texto del vínculo.

```
<krpointer
uri="http://purl.org/NET/durito/ontologies/2005/09/repository#at_59E"
autotext="true"/>
```

Y esto desplegará un vínculo al mismo objeto describable, pero con el texto "esta fotografía":

```
<krpointer
uri="http://purl.org/NET/durito/ontologies/2005/09/repository#at_59E">
esta fotografía</krpointer>
```

`br`

Elemento vacío que señala un salto de línea forzado.

`em`

Señala texto enfatizado, el cual normalmente será presentado con letras cursivas.

`hi`

Señala texto que debe enfatizarse para una función especial (por ejemplo, texto encontrado en un resultado de búsqueda).

Aquí un ejemplo de un valor de este tipo en [repository](#):

```

rep:archival_thing_23    a swv:Image ;
                           swv:hasNotes "Ésta es la única imagen en existencia de la
especie
                           <em>ardillus sucius coyocanensus</em>. Al parecer fue
tomada con la misma
                           cámara que la foto
                           <krpointer
uri='http://purl.org/NET/durito/ontologies/2005/09/repository#archival_thing_22'
autotext='true' />."@es^^sys:LFMarkup .

```

Los ComplexDataType

Las especificaciones de la WS también sientan las bases de los **ComplexDataType** al definir los "valores estructurados", según se llaman en dichos documentos. En estas estructuras siempre existe un nodo raíz del cual los demás triples que constituyen el valor se desprenden en forma de árbol. En un triple del RC una propiedad que señale un valor de un **ComplexDataType** tiene como objeto dicho nodo raíz, el cual *no* constituye un objeto descriptible de RC.

Los mecanismos que usa el Pescador para los **ComplexDataType** van más allá de las especificaciones de la WS en tanto permiten definir reglas de validación para capturar los valores y establecen con mayor claridad el significado de estas estructuras, lo cual posibilita diversos tipos de procesamiento automático.

En el RC se señala que una subgráfica representa el valor de un **ComplexDataType** por medio de la clase del nodo raíz de dicha subgráfica.

Por ejemplo, en el archivo [shared-worldview.def](#), se define `swv:TimeRangeWSE` ("rango de tiempo con los punto de inicio y final especificados") como un tipo de datos de la categoría **ComplexDataType**. Luego, en [repository](#) se emplea un valor de este tipo de datos para señalar que una fotografía fue tomada en algún momento entre 1950 y 1952 (renglones 327 a 324):

```

rep:archival_thing_23    a swv:Image ;
                           (...)
                           swv:hasInitialPhotoDate [    a swv:TimeRangeWSE;
                                                           swv:starts
"1950"^^swv:YearInCommonEra ;
                                                           swv:ends
"1952"^^swv:YearInCommonEra ] ;

```

En este caso existe un [descriptor](#) cuyos sujeto, predicado y objeto son, respectivamente, `rep:archival_thing_23`, `swv:hasInitialPhotoDate` y el nodo anónimo que es el nodo raíz de un valor con el tipo de datos `swv:TimeRangeWSE`. Esta parte de la gráfica "quiere decir" que el valor de la propiedad `swv:hasInitialPhotoDate` para `rep:archival_thing_23` se construye por medio del árbol de afirmaciones que se desprenden del nodo anónimo, según las reglas que establece la definición del mencionado tipo de datos. (Dicha definición se encuentra en el archivo [shared-worldview.def](#).)

Este valor se desplegaría ante el usuario como "1950 - 1952". (Véase la sección [Funciones de valores de base para la presentación](#) para una explicación de cómo se construiría dicho texto.) Es de notar que los triples que se desprenden del nodo anónimo también son mapeados por [descriptores](#).

(Desde la perspectiva de la representación de los significados por medio de las ontologías, una estructura de tipo **ComplexDataType** desempeña la misma función que un literal. En teoría nunca debe ser necesario referir "fuera de contexto" al nodo raíz de una estructura de esta categoría--es decir, nunca debe haber porqué incluirlo en otros triples, más allá del triple que lo tiene como objeto y lo conecta con un objeto descriptible, y los que lo emplean como sujeto para plasmar el valor en sí. Por este motivo es apropiado que sea un nodo anónimo.)

Los AbstractDataType

Como ya se mencionó, los tipos de datos de la categoría `AbstractDataType` nunca pueden asignarse directamente a un valor en el RC. Más bien, su función es agrupar a otros tipos de datos por medio de una estructura jerárquica. Por ejemplo, en el archivo [shared-worldview.def](#), se define el tipo de datos `swv:TimeRange` como un `AbstractDataType`, cuyos descendientes son dos tipos de datos de la categoría `ComplexDataType`: `swv:TimeRangeWSE` y `swv:TimeRangeOverUnit`--dos formas distintas de especificar un rango de tiempo. (El primero, `swv:TimeRangeWSE`, se usa para un rango tipo "1950-1952", cuyo significado preciso es "del primero de enero de 1950 al 31 de diciembre de 1952"; el segundo, `swv:TimeRangeOverUnit`, es para rangos que sólo mencionan una unidad del calendario, por ejemplo "1950", que significa "del primero de enero de 1950 al 31 de diciembre de 1950".

El `AbstractDataType` `swv:TimeRange` no se asigna a valor alguno en [repository](#), pero sí se establece, en [shared-worldview.def](#) como rango de la propiedad `swv:hasInitialPhotoDate` (indirectamente, por medio de la superpropiedad, `swv:hasRelatedEventTime`). Por lo tanto, a los valores de esta propiedad se les puede asignar cualquier tipo de datos descendiente de `swv:TimeRange`, con la condición de que sea un tipo de datos de la categoría `ComplexDataType` o `FundamentalDataType`. (Véase, en [repository.n3](#), los renglones 311 a 312, 320 a 321 y 332 a 334.)

Los vectores de comparación

Los vectores de comparación proporcionan mecanismos especiales para comparar valores. Esto permite ordenar correctamente los conjuntos de objetos descriptibles--incluso con base en valores de diversos tipos de datos.

Considere por ejemplo los sistemas de clasificación (como Dewey Decimal o Library of Congress) que a menudo integran letras y números bajo un esquema *sui generis*. Resulta que para ordenar las clasificaciones se requiere un algoritmo especial. Los vectores de comparación permiten, en situaciones como ésta, establecer una función (por medio de un [bloque de código Ruby](#)) que implemente el algoritmo apropiado.

Otro ejemplo son los tipos de datos abstractos cuyos descendientes son tipos de datos complejos que presentan una diversidad de estructuras internas. En algunos casos no es apropiado comparar los valores (nos referimos a los valores que aparecen el RC) a partir de los [valores de base para la presentación de texto](#) generados; se necesitan funciones especiales para implementar la comparación. De nuevo, los vectores de comparación permiten definir dichas funciones.

Véase la sección [Archivos de definiciones](#) para la explicación concreta acerca de cómo emplear este mecanismo.

Reglas de herencia en los tipos de datos

Son sencillas las reglas de herencia para los tipos de datos:

1. A los valores del tipo de datos *X* se les asigna implícitamente todos los tipos de datos del conjunto *Y*, donde *Y* es el conjunto de tipos de datos ancestros de *X*.
2. Se permite la herencia múltiple entre los tipos de datos siempre y cuando no existan ambigüedades acerca qué `RuleSet` se vincule con cada tipo de datos de las categorías `ComplexDataType` y `FundamentalDataType`. (Véase [Conjuntos de reglas](#) y, más

adelante, la explicación de cómo vincular un **RuleSet** con un tipo de datos determinado y cómo esta vinculación puede heredarse de un tipo de datos a otro a través de la jerarquía.)

Mecanismos de captura

En el proceso de modificación del RC, la vista global recibe datos del usuario para su integración al RC. Por lo tanto, la vista global debe establecer mecanismos de captura específicos para todos los tipos de datos de las categorías **FundamentalDataType** y **ComplexDataType**. (Aquí nos referimos a elementos de la interfaz de usuario que proporcionen a éste formas amigables de ingresar información al sistema. Son otra cosa muy distinta las reglas de validación de datos que mencionamos arriba, las cuales forman parte de la capa de lógica de RC y ayudan a asegurar la coherencia de los contenidos del RC.)

Relaciones de RC

Las relaciones de RC ([KRRelation](#)) son un mecanismo para organizar y simplificar el movimiento a través de segmentos de la gráfica RDF; facilitan la tarea de configurar la lógica de RC. Además permiten que el sistema efectúe anticipadamente los pasos de diversas operaciones, lo cual acelera su realización. En esencia, las relaciones de RC establecen una conexión entre un nodo RDF y otro. Son direccionales: se considera que uno de los nodos es la “entrada” y el otro es la “salida”. Existen dos tipos de relaciones de RC:

SingleDescriptorRelation

El tipo de relación básico. Atraviesa la gráfica RDF desde el sujeto de un descriptor hasta objeto del mismo.

MultipleDescriptorRelation

Atraviesa un camino compuesto por múltiples descriptors en cadena. Empieza en el sujeto del primer descriptor y termina en el objeto del último descriptor.

Las relaciones de RC se establecen por medio de reglas denominadas [KRRelationRule](#), que determinan la ruta que se atraviesa en la gráfica RDF.

Considere el siguiente fragmento de la ontología [repository](#):

```
rep:archival_thing_21    a swv:Image ;
                        (...)
                        swv:hasTopic [          a sys:MultOrd0bjectsSeq;
                                                rdf:_1 rep:archival_thing_8 ] .

rep:archival_thing_22    a swv:Image ;
                        (...)
                        swv:hasTopic [          a sys:MultOrd0bjectsSeq;
                                                rdf:_1 rep:archival_thing_8;
                                                rdf:_2 rep:archival_thing_9;
                                                rdf:_3 rep:archival_thing_10 ] .
```

En el archivo de definiciones [archivo-juan-perez.def](#), en el **RuleSet** **image_as_content** se establece una regla que genera relaciones de RC del tipo **SingleDescriptorRelation**. Dicha regla se llama **has_topic** y relaciona el sujeto y el objeto de todos los descriptors que tienen como predicado la propiedad **swv:hasTopic** y como sujeto un miembro de la colección vinculada con este **RuleSet** (es decir, un miembro de la colección para la organización del sitio **rep:soc_ajp_images_as_content**).

Entonces, por ejemplo, la lógica de RC permite solicitar los nodos de salida de todas las relaciones

generadas por la regla `has_topic` que tienen como entrada `rep:archival_thing_22`. La respuesta que se recibiría al hacerlo es el conjunto ordenado de nodos `rep:archival_thing_8`, `rep:archival_thing_9` y `rep:archival_thing_10`.

Por otra parte las relaciones de RC son invertibles: se puede partir del nodo de salida y terminar en el nodo de entrada. Por lo tanto, se pueden solicitar los nodos de entrada de todas las relaciones generadas por `has_topic` que tienen como nodo de salida `rep:archival_thing_8`. Al hacer esta solicitud se recibe como respuesta el conjunto no ordenado de nodos `rep:archival_thing_21` y `rep:archival_thing_22`.

Con la finalidad de explicar el funcionamiento de las relaciones de RC y las reglas que las definen, exponemos de manera resumida el sintaxis que se emplea en los archivos de definiciones para referirlas.

`[[abreviación del archivo de definiciones.]nombre del RuleSet.]nombre de la regla de relación de RC`

Los corchetes indican las partes opcionales de la expresión. Se puede omitir la abreviación del archivo de definiciones si se trata de una regla establecida por un `RuleSet` del mismo archivo donde aparece la expresión. Y también se puede omitir el nombre del `RuleSet` si se está hablando de una regla establecida dentro del mismo `RuleSet` donde aparece la expresión.

Por lo tanto, desde cualquier `RuleSet` del archivo [archivo-juan-perez.def](#) se puede hablar de la regla `has_topic`, definida en el `RuleSet` `image_as_content`, de la manera siguiente:

```
image_as_content.has_topic
```

Una inversión de dicha regla es igual de sencilla de representar:

```
image_as_content.has_topic.inv
```

Para conectar una [KRRelationRule](#) con un nodo de salida o entrada concreta, o con la salida o entrada de otra `KRRelationRule`, se emplea el operador `->`.

Este mecanismo, a la hora de emplearse en las `MultipleDescriptorRelation`, simplifica enormemente la descripción de caminos complicados en la gráfica RDF.

Por ejemplo, considere los vínculos complejos que se establecen en los datos de ejemplo entre objetos físicos y las obras intelectuales que se plasman en dichos objetos físicos. En [repository](#) para pasar de un `swv:PhotographicPrint` (un objeto físico) al título de la `swv:Image` (una obra intelectual) que dicho objeto plasma, es necesario atravesar ¡tres triples!, tal como muestran los siguientes fragmentos de dicha ontología.

```
rep:archival_thing_18    a swv:PhotographicPrint;
                        swv:hasMainContentLink rep:archival_thing_12 .
```

```
rep:archival_thing_12    a swv:ContentLink;
                        swv:hasLinkedContent rep:archival_thing_21 .
```

```
rep:archival_thing_21    a swv:Image;
                        swv:hasTitle "La Ciudad de México desde la
catedral"@es^xsd:string .
```

Sin embargo, en el `RuleSet` `photographic_print` del archivo [archivo-juan-perez.def](#) se representa este camino de una manera sencilla y (creemos) intuitiva:

```
has_main_content->image_as_content.has_title
```

En este caso, `has_main_content` es en sí una regla que genera relaciones del tipo `MultipleDescriptorRelation`. El `RuleSet` `photographic_print` retoma esta regla

del archivo [shared-worldview.def](#). Aquí el fragmento de [archivo-juan-perez.def](#) que establece la regla `has_main_content`:

```
<RuleSet photographic_print>
  (...)
  <KRRRelation has_main_content>
    swv.manifestation.has_main_content
    (...)
  </KRRRelation>
  (...)
</RuleSet>
```

En [shared-worldview.def](#), el RuleSet `manifestation` define la regla `has_main_content` de la manera siguiente:

```
<RuleSet manifestation>
  (...)
  <KRRRelation has_main_content>
    has_main_content_link->content_link.has_linked_content
  </KRRRelation>
  (...)
</RuleSet>
```

Nótese que el RuleSet `manifestation` construye esta regla a partir de dos reglas que generan relaciones de tipo `SingleDescriptorRelation`, la primera definida en el mismo RuleSet y la segunda, retomada desde otro RuleSet llamado `content_link`.

Con este mecanismo se logra un nivel importante de modularidad en la definición de los caminos en la gráfica RDF y se evita la repetición de definiciones, lo cual vuelve el sistema más flexible y hace que el sintaxis para definir dichos caminos sea más leíble.

(En el apartado [Archivos de definiciones](#) se describe con precisión la manera en que se establecen las relaciones de RC por medio de los archivos de definiciones.)

Valores de base para la presentación

Los valores de base para la presentación ([BPV](#)) son los elementos básicos de contenido de las [descripciones](#). Por el momento existen dos tipos de BPV:

`textBPV`

Como señala el nombre, la finalidad de este objeto es representar un valor de texto. Sin embargo, cuando el valor sale de la capa de lógica de RC como parte de una descripción en formato de base para la presentación, se registra dicho texto como un fragmento bien equilibrado de XML. Las etiquetas que puede contener son las mismas que el tipo de datos [LFMarkup](#), excepto que las etiquetas `krpointer` no pueden ser elementos vacíos y no pueden tener un atributo `autotext`.

`imageBPV`

Este objeto representa una imagen digital. Puede ser enviada por la capa de lógica de RC sin indicador de tamaño--en dado caso la vista global podrá modificar el tamaño de la imagen según sus necesidades. Pero también puede ser enviada con un indicador de tamaño. Los indicadores disponibles son `thumb`, `fitViewport` o `full`; permiten imponer distintas restricciones sobre el tamaño y resolución con el cual la vista global despliega la imagen.

Funciones de valores de base para la presentación

Las funciones de valores de base para la presentación ([BPVFunction](#)) generan los BPV. Existen dos tipos de BPVFunction:

`textBPVFunction`

Produce un `textBPV`.

`imageBPVFunction`

Produce un `imageBPV`.

Las BPVFunction normalmente generan un BPV a partir de un nodo RDF. Constituyen el mecanismo de traducción entre los nodos RDF y los componentes de las descripciones. Todos los nodos RDF que puedan fungir de objetos de un descriptor y todos los objetos descriptibles tienen una `textBPVFunction default`, que constituye el mecanismo predeterminado de representar el nodo en las descripciones.

En algunos casos--por ejemplo, los nodos literales del tipo `xsd:string`--la `textBPVFunction` predeterminada traslada directamente el contenido del literal al `textBPV`. Para representar un objeto descriptible de RC, a menudo la `textBPVFunction` predeterminada atraviesa la gráfica RDF por medio de una relación de RC, para llegar a un nodo literal determinado, cuyo contenido se vuelve el `textBPV`. También es posible elaborar los `textBPV` con base en algoritmos más complejos; para ello se permite la definición de bloques de código Ruby.

Por ejemplo, ésta es la `textBPVFunction` predeterminada que establece el `RuleSet photographic_print` (en el archivo [archivo-juan-perez.def](#)) para generar una representación textual de los miembros de la colección `rep:soc_ajp_photographic_prints`:

```
(ROOT->has_unique_id).textBPV
```

Aclaremos que en los archivos de definiciones se emplea la palabra clave [ROOT](#), que refiere automáticamente el objeto descriptible o el nodo raíz de un `ComplexDataType` vinculado al `RuleSet` en cuestión. La expresión entre paréntesis constituye una solicitud de búsqueda de todas las relaciones que tienen como entrada el nodo representado por `ROOT` y que fueron generados por la regla `has_unique_id`. Aquí el código presupone que esta búsqueda encontrará una sola relación. El método `textBPV` genera el valor de presentación de base en formato de texto correspondiente al nodo de salida de la relación encontrada.

Considere entonces el objeto descriptible `rep:archival_thing_18`, miembro de la colección `rep:soc_ajp_photographic_prints` en la ontología [repository](#):

```
rep:archival_thing_18    a swv:PhotographicPrint;  
                        swv:hasUniqueIdentifier "MICH01"^^xsd:string .
```

Para crear una representación textual de este objeto, el sistema busca la relación de RC que generó la regla `has_unique_id` y que tiene como nodo de entrada `rep:archival_thing_18` y pide el nodo de salida de dicha relación; así encuentra el literal "MICH01", y ejecuta la `textBPVFunction` predeterminada de éste nodo, obteniendo un `textBPV` con el contenido de dicho literal.

A continuación un ejemplo que emplea un bloque de código Ruby. Se trata de la `textBPVFunction` predeterminada que establece el `RuleSet person`:

```
{ (ROOT->has_last_names).textBPV + ", " + (ROOT->has_first_names).textBPV }
```

Esta función podría ejecutarse sobre el nodo `rep:archival_thing_1`:

```
rep:archival_thing_1    a swv:Person ;  
                        swv:hasGivenNames "Julio"^^xsd:string;  
                        swv:hasLastNames  "Michaud"^^xsd:string .
```

El resultado sería un `textBPV` con el texto "Michaud, Julio".

Fragmentos de descripciones

Los fragmentos de descripciones ([DescFragment](#)) son elementos de las descripciones ligeramente más complejos que los BPV. Normalmente consisten en un `textBPV` que funciona como "etiqueta" y un `textBPV` o `imageBPV` que funciona como "valor". Aquí se trata de el patrón "etiqueta-valor" estándar de las fichas catalográficas, que a menudo se presenta en una tabla de dos columnas.

El formato de presentación concreto de los `DescFragment` (por ejemplo, en una tabla, o sin una y con un formato de letra diferente para resaltar la etiqueta) dependerá del medio de presentación, el contexto en el cual se emplea la descripción, y la ubicación del `DescFragment` en relación con los demás elementos de dicha descripción. La decisión final acerca de cómo presentar un `DescFragment` determinado será la responsabilidad de la vista global.

Las reglas para crear fragmentos de descripción concretos se llaman [DescFragmentTemplate](#). Dichas reglas pueden generarse automáticamente por medio de un algoritmo predeterminando, o pueden configurarse explícitamente por medio de los archivos de definiciones. (Véase [df](#).)

Tipos y configuración de descripciones

Ya expuestos los puntos anteriores, podemos presentar algunos aspectos adicionales del funcionamiento de las descripciones.

Existen cuatro tipos de descripción:

- descripciones completas (`FullDesc`);
- descripciones breves (`ShortDesc`);
- descripciones variables (`VariableDesc`); y
- descripciones para la modificación de datos.

La lógica de RC establece estos tipos de descripción en vistas de las diversas situaciones en que el sistema Pescador describe los objetos descriptibles. `FullDesc` se presenta al usuario cuando éste solicita ver toda la información disponible acerca de un objeto determinado; `ShortDesc` se usa en situaciones donde, por razones de espacio, se requiere de una descripción resumida de un objeto (como en los listados); `VariableDesc` es una descripción cuyos contenidos se eligen de manera dinámica (se emplea, por ejemplo, en los resultados de búsqueda, donde para cada objeto encontrado se muestran únicamente los campos que coinciden con los criterios de búsqueda); y finalmente, las descripciones para la modificación de datos son descripciones que incluyen controles que permiten su modificación.

Los archivos de definiciones proporcionan mecanismos para configurar el contenido y estructura de los tres primeros tipos de descripción (`FullDesc`, `ShortDesc` y `VariableDesc`) mientras que el contenido y estructura de las descripciones para la modificación de datos se determinan automáticamente por medio de un algoritmo que toma en cuenta tanto la configuración de la `FullDesc` como las reglas establecidas para la estructuración del RC.

Las `FullDesc` y `ShortDesc` se configuran por medio de plantillas con código encrustado (similares a las plantillas para página Web de sistemas como JSP, ASP, PHP y ERB, entre otros). Estas plantillas consisten en elementos fijos que se trasladan tal cual a la descripción, y bloques de código

encrustados, cuya función es generar elementos para su inclusión en la descripción o modificar el flujo de procesamiento de la plantilla. Los elementos fijos o generados por código que pueden aparecer en las plantillas incluyen los [DescFragment](#), [BPV](#) y componentes estructurales como áreas y bloques.

Las configuraciones de las **VariableDesc** consisten en un listado de **DescFragments** que el sistema puede seleccionar para su inclusión en las **VariableDesc**, según las necesidades del momento.

En el apartado [Elementos](#) se explica detalladamente cómo configurar las descripciones por medio de los archivos de definiciones.

A continuación un ejemplo de una plantilla de descripción que aparece en el **RuleSet** **photographic_print** del archivo [archivo-juan-perez.def](#). La función de esta plantilla es generar descripciones de impresiones fotográficas. Crea una área principal con título y un bloque de **DescFragments** y, en el caso de existir una copia digital de la fotografía, la incluye en una segunda área de la descripción.

```
<FullDesc>
  <Area main>
    <Title>
      "Full record"@en
      "Ficha completa"@es
    </Title>
    <Block>
      df has_unique_id
      df has_title
      df has_description
      df has_initial_photo_location
      df has_initial_photo_date
      df has_topic
      df has_notes
      df has_image_media_type
      df has_conservation_status
      <% if ((ROOT->has_system_accessible_digital_copy).hasObj)
%>
        <UniqueDF>
          "Digital copy"@en
          "Copia digital"@es
          uniqueImageBPV { (ROOT-
>has_system_accessible_digital_copy).imageBPV.thumb }
          <linkTo image>
            "View in higher resolution..."@en
            "Versión con mayor resolución..."@es
          </linkTo>
        </UniqueDF>
      <% end %>
    </Block>
  </Area>
  <% if ((ROOT->has_system_accessible_digital_copy).hasObj) %>
    <Area image>
      <Title>
        "Image"@en
        "Imagen"@es
      </Title>
      uniqueImageBPV { (ROOT-
>has_system_accessible_digital_copy).imageBPV.fitViewPort }
      df has_unique_id
      df has_title
    </Area>
  <% end %>
</FullDesc>
```

Operaciones

A continuación se describen las diversas operaciones que lleva a cabo la capa de lógica de RC.

Con respecto al diseño inicial del código: son las clases del paquete

[mx.org.pescador.krmodel.operations](#)--en concreto las subclases de [Operator](#)--las que llevan a cabo las operaciones de lógica de RC. Los objetos que se generan como resultados de una operación son subclases de [KRViewObj](#), ubicadas en el paquete [mx.org.pescador.krview](#).

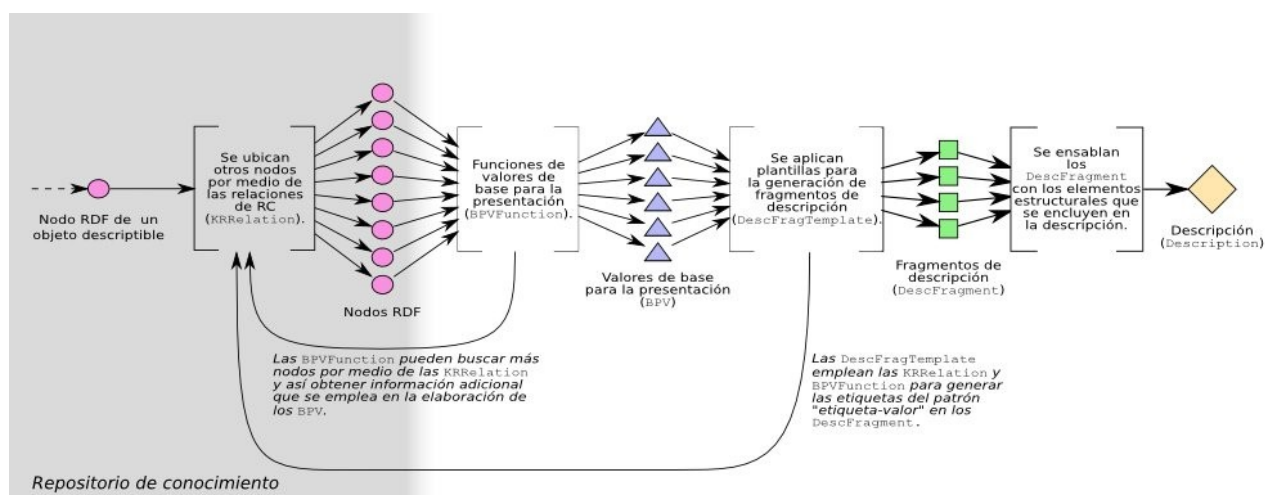
Extracción de datos

Descripciones

Cualquier descripción menos las descripciones para la modificación de datos

Clase de objeto generado	Clase de objeto que lo genera	Uso
Description	DescriptionCreator	Cualquier contexto en el cual se presenta una ficha de un objeto descriptible; véase la explicación general y la descripción de tipos y configuración de las descripciones.
Algoritmo		
Los detalles de cómo llevar a cabo este proceso son determinados por un DescTemplate . • Se parte del nodo RDF del objeto descriptible. • Se verifican los permisos de visualización del objeto descriptible. • Por medio de las KRRelation se buscan otros nodos relacionados con el primero. • Se llevan a cabo las BPVFunction para generar, con base en dichos nodos, los BPV. • Se aplican las DescFragmentTemplate para generar los DescFragment, los cuales contienen pares de etiquetas y valores. • Se ensamblan los DescFragment con los elementos estructurales que incluye la descripción.		

La siguiente gráfica sintetiza este proceso:



Descripciones para la modificación de datos

Clase de objeto generado	Clase de objeto que lo genera	Uso
DescriptionForMod	DescriptionCreator	La modificación de la información descriptiva asociada a un objeto descriptible; véase la explicación general y la descripción de tipos y configuración de las descripciones.

Algoritmo

Los detalles de cómo llevar a cabo este proceso son determinados por un [DescTemplate](#).

- Se parte del nodo RDF del objeto descriptible.
- Se verifican los permisos de modificación del objeto descriptible.
- Por medio de las [RRRelation](#) se buscan otros nodos relacionados con el primero y se generan los [ModPosElement](#) relevantes.
- Se llevan a cabo las [BPVFunction](#) para generar, con base en dichos nodos, los [BPV](#) con los [ModPosElement](#) incrustados.
- Se aplican las [DescFragmentTemplate](#) para generar los [DescFragment](#), los cuales contienen pares de etiquetas y valores.
- Se ensamblan los [DescFragment](#) con los elementos estructurales que incluye la descripción, incluyendo algunos [ModPosElement](#) vinculados con el objeto descriptible en general.

(En resumen, se siguen los mismos pasos que para los [otros tipos de descripción](#), salvo que en diversas partes de la descripción se incrustan los [ModPosElement](#) que ofrecen la posibilidad de modificar de la gráfica RDF.)

Generación de conjuntos

La capa de lógica de RC puede crear diversos tipos de conjuntos--todos ellos subclases de [Set](#). Tienen la tarea de generarlos las subclases de [SetCreator](#).

La lógica de RC ofrece diversos mecanismos para generar conjuntos de nodos referidos a objetos descriptibles:

- Membresía del objeto descriptible en un [PersistentGroup](#)
- Búsqueda de texto libre en los [TextBPV](#)
- Membresía del objeto que se presente como nodo de salida de una relación de RC, en un conjunto determinado
- Nodos únicos existentes para las salidas de relaciones de RC generadas por una regla determinada
- Clase (tipo) del objeto descriptible
- Valor o rango de valores
- Referencia temporal
- Referencia espacial
- Intersección de dos o más conjuntos generados con base en los criterios anteriores
- Unión de dos o más conjuntos generados con base en los criterios anteriores

Conjuntos ordenados de descripciones

Clase de objeto generado	Clase de objeto que lo genera	Uso
OrderedDescSet	OrderedDescSetCreator	• Listado de los miembros de una colección • Listado de todas las colecciones en el sitio • Listado de áreas de repositorio • Resultados de búsqueda • Listado de los miembros de un grupo en un GroupedDescSet
Algoritmo		
• Se seleccionan los nodos de los DescribableObject del conjunto y se verifican los permisos de visualización. • En algunos casos se analiza el conjunto para establecer las posibilidades de ordenamiento del conjunto y el orden predeterminado del mismo. • Se ordenan los nodos. • Se genera una división del conjunto en páginas. • Se generan las Description.		

Conjuntos ordenados de descripciones para operaciones de modificación de datos

Clase de objeto generado	Clase de objeto que lo genera	Uso
OrderedDescSetForMod	OrderedDescSetCreator	Mover, borrar o crear los objetos descriptibles que forman parte de

		una colección o área de repositorio
Algoritmo		
• Se seleccionan los nodos de los DescribableObject del conjunto y se verifican los permisos de modificación. • En algunos casos se analiza el conjutno para establecer las posibilidades de ordenamiento del conjunto y el orden predeterminado del mismo. • Se ordenan los nodos. • Se genera una división del conjunto en páginas. • Se generan las Description. • Se generan los ModPosElement.		

Conjuntos ordenados de BPV

Clase de objeto generado	Clase de objeto que lo genera	Uso
OrderedBPVSet	OrderedBPVSetCreator	• Listado de opciones para fijar el valor de determinados campos • Listado de opciones para elegir criterios de búsqueda
Algoritmo		
• Se seleccionan los nodos de los DescribableObject del conjunto y se verifican los permisos de visualización. • En algunos casos se analiza el conjutno para establecer las posibilidades de ordenamiento del conjunto y el orden predeterminado del mismo. • Se ordenan los nodos. • Se genera una división del conjunto en páginas. • Se generan los BPV.		

Estructuras de subgrupos

Clase de objeto generado	Clase de objeto que lo genera	Uso
GroupingStructure	GroupingStructureCreator	La división de un Set en páginas
Algoritmo		
• Se seleccionan los nodos de cada subgrupo. • Se ordenan los nodos y los subgrupos. • Se genera un TextBPV que describe cada subgrupo.		

Conjuntos de descripciones en subgrupos

Clase de objeto generado	Clase de objeto que lo genera	Uso
GroupedDescSet	GroupedDescSetCreator	La división de un OrderedDescSet en subgrupos con base en los nodos de salida de las relaciones de RC generadas por una regla determinada ("Navegación facetada").
Algoritmo		
• Se seleccionan los nodos de cada subgrupo. • Se ordenan los nodos y los subgrupos. • Se genera un TextBPV que describe cada subgrupo. • Se generan las Description.		

Conjuntos de descripciones en jerarquía

Clase de objeto generado	Clase de objeto que lo genera	Uso
HierarchicalDescSet	HierarchicalDescSetCreator	Visualización de una colección que se organice en una jerarquía (como un tesoro).
Algoritmo		
• Se seleccionan los nodos de los DescribableObject del conjunto y se verifican los permisos de visualización. • Se establece la estructura jerárquica. • Se generan las Description.		

Conjuntos de descripciones en jerarquía para operaciones de modificación de datos

Clase de objeto generado	Clase de objeto que lo genera	Uso
HierarchicalDescSetForMod	HierarchicalDescSetCreator	Mover, borrar o crear los objetos descriptibles que forman parte de una colección que se organice en una jerarquía (como un tesoro).
Algoritmo		

- Se seleccionan los nodos de los [DescribableObject](#) del conjunto y se verifican los permisos de modificación.
- Se establece la estructura jerárquica.
- Se generan las [Description](#).
- Se generan los [ModPosElement](#).

Conjuntos de BPV en jerarquía

Clase de objeto generado	Clase de objeto que lo genera	Uso
HierarchicalDescSet	HierarchicalBPVSetCreator	• Listado de opciones para fijar el valor de determinados campos, cuando se trate de una colección que se organice en una jerarquía (como un tesauo) • Listado de opciones para elegir criterios de búsqueda, cuando se trate de una colección que se organice en una jerarquía (como un tesauo)
Algoritmo		
• Se seleccionan los nodos de los DescribableObject del conjunto y se verifican los permisos de visualización. • En establece la estructura jerárquica. • Se generan los BPV.		

Modificación de datos

Generar un objeto descriptible

Clase de objeto que ofrece la posibilidad de modificación	Clase de objeto que lleva a cabo la modificación
D0CreateModPos	D0Creator
Algoritmo	
• Se verifican los permisos de modificación. • Se genera una DescriptionForModTemplate apropiada.	

- Se establece un [LockSet](#) apropiado.
- Se genera y se envía al controlador global una [DescriptionForMod](#).
- Se reciben los datos aportados por el usuario por medio de los [ModPOsElement](#).
- Si la nueva estructura propuesta para el RC cumple con todas las [StructureRule](#) establecidas, entonces se genera un nuevo [DescribableObject](#) junto con la gráfica RDF que lo describe.
- Se suprime el [LockSet](#).

Borrar un objeto descriptible

Clase de objeto que ofrece la posibilidad de modificación	Clase de objeto que lleva a cabo la modificación
D0DeleteModPos	D0Deleter
Algoritmo	
• Se verifican los permisos de modificación. • Se establece un LockSet apropiado. • Si al borrar el DescribableObject junto con la gráfica RDF que lo describe no violaría ninguna StructureRule establecida, entonces se procede a borrarlo. • Se borra el LockSet.	

Mover un objeto descriptible

Clase de objeto que ofrece la posibilidad de modificación	Clase de objeto que lleva a cabo la modificación
D0MoveModPos	D0Mover
Algoritmo	
• Se verifican los permisos de modificación. • Se establece un LockSet apropiado. • Si el movimiento del DescribableObject solicitado requiere modificar la gráfica RDF que lo describe, se genera una DescriptionForModTemplate especial, se establece un nuevo LockSet, se genera y se envía al controlador global una DescriptionForMod, y se reciben los datos aportados por el usuario por medio de los ModPOsElement. • Si la nueva estructura propuesta para el RC cumple con todas las StructureRule establecida, entonces se procede a mover el DescribableObject (y en dado caso, generar las nuevas estructuras que lo describen). • Se suprimen los LockSet.	

Modificar una relación de RC

Clase de objeto que ofrece la posibilidad de modificación	Clase de objeto que lleva a cabo la modificación
---	--

KRRelationModPos	KRRelationModifier
Algoritmo	
• Se verifican los permisos de modificación. • Se establece un LockSet apropiado. • Si la nueva relación de RC permite cumplir con todas las StructureRule, se procede a la modificación. • Se suprime el LockSet.	

Modificar simultáneamente múltiples relaciones de RC

Clase de objeto que ofrece la posibilidad de modificación	Clase de objeto que lleva a cabo la modificación
MultKRRelationModPos	MultKRRelationModifier
Algoritmo	
• Se verifican los permisos de modificación. • Se establece un LockSet apropiado. • Si las nuevas relaciones de RC permiten cumplir con todas las StructureRule, se procede a la modificación. • Se suprime el LockSet.	

Verificación de estructuras

Esta operación es llevada a cabo por la clase [GraphStructureChecker](#). Verifica lo siguiente:

- Que todos los [DescribableObject](#) pertenezcan a exactamente una colección para la organización del sitio ([SOC](#)).
- Que todas las colecciones para la organización del sitio pertenezcan a exactamente una área del repositorio ([RepositoryArea](#)).
- Que la gráfica cumpla con todas las [StructureRule](#) establecidas.

Archivos de definiciones

Los archivos de definiciones son el mecanismo fundamental de configuración de la lógica de RC. A continuación se explica de manera detallada su funcionamiento.

Sintaxis

Se ha retomado el sintaxis básico de los archivos de configuración del servidor http Apache. Se establecen las secciones de los archivos de definiciones por medio de etiquetas de inicio y de terminación, y en determinadas etiquetas de inicio se incluye una cadena que identifica la sección.

Asimismo las secciones pueden contener subsecciones o directrices con cero, uno o varios argumentos, según el caso.

Aquí un ejemplo de una sección, que es de tipo [RuleSet](#) y se llama `group_link`:

```
<RuleSet group_link>
  (... directrices o subsecciones ...)
</RuleSet>
```

Cada etiqueta y directriz debe colocarse en un renglón nuevo. Se ignora el espacio en blanco.

Resumen

Los archivos de definiciones siempre inician con una sección [Realm](#) que establece parámetros generales tales como el URI de la ontología generada y el nombre amigable del área de repositorio generada. Aquí un ejemplo de una sección [Realm](#):

```
<Realm>
  "Archivo Juan Pérez"@es
  "Juan Pérez Archive"@en

  abbreviation ajp
  thisOntology "http://archive.specific.uri/archivo-juan-perez#"

  dependsRealm system
  dependsRealm shared-worldview

  <RepositoryArea>
    "Archivos de fotografías Juan Pérez"@es
    "Juan Pérez Photo Archive"@en
    generalArchivalThings
  </RepositoryArea>
</Realm>
```

Después aparecen una o más secciones [DefinitionSet](#), que permiten agrupar las definiciones en bloques lógicos. Al interior de un [DefinitionSet](#) se puede incluir cero, una o más secciones [Vocabulary](#), que definen clases, propiedades y tipos de datos, y cero, una o más secciones [RuleSet](#), por medio de las cuales se configuran la mayoría de las operaciones de lógica de RC en sí. Finalmente, en los [DefinitionSet](#) pueden incluirse cero, una o más secciones [RepositorySetup](#), que establecen la presencia de determinados elementos en el área de repositorio que genera el archivo de definiciones.

A continuación un ejemplo de un [DefinitionSet](#) sencillo:

```
<DefinitionSet describing_people>
  <Vocabulary>
    <Class swv:Person>
      "Person"@en
      "Persona"@es
    </Class>

    <Property swv:hasGivenNames>
      "Given Names"@en
      "Nombres"@es
      subPropertyOf swv:hasName
      range xsd:string
    </Property>

    <Property swv:hasLastNames>
```

```

        "Last Names"@en
        "Apellidos"@es
        subPropertyOf swv:hasName
        range xsd:string
    </Property>
</Vocabulary>

<RuleSet person>
    <Structure>
        <Descriptor has_given_names>
            property swv:hasGivenNames
            minCardinality 1
            maxCardinality 1
            outRequirement fromPropRange
        </Descriptor>
        <Descriptor has_last_names>
            property swv:hasLastNames
            minCardinality 1
            maxCardinality 1
            outRequirement fromPropRange
            defaultOrderBy
        </Descriptor>
    </Structure>

    <BPVFunctions>
        <TextBPVFunction full_name_fl>
            orderUsing has_last_names
            { (ROOT->has_given_names).textBPV + " " + (ROOT-
>has_last_names).textBPV }
        </TextBPVFunction>

        <TextBPVFunction full_name_lf>
            defaultTextBPVFunction
            { (ROOT->has_last_names).textBPV + ", " + (ROOT-
>has_first_names).textBPV }
        </TextBPVFunction>
    </BPVFunctions>

    <Descriptions>
        VariableDesc default
        ShortDesc default
        FullDesc default
    </Descriptions>
</RuleSet>

<RepositorySetup>
    <SOC people>
        "People mentioned in collecitions in these archives"@en
        "Personas mencionadas en las colecciones de estos archivos"@es
        hasGroupDomain swv:Person
        bindToRuleSet person
        secondaryLevel
    </SOC>
</RepositorySetup>
</DefinitionSet>

```

El objetivo de este **DefinitionSet** es configurar el procesamiento de la información que describe a las personas. Primero en la sección [Vocabulary](#) se define la clase `swv:Person` y dos propiedades para describirlo. Después aparece una sección [RuleSet](#) que fija cómo debe estructurarse la subgráfica RDF que describe a las personas y cómo debe procesarse la información plasmada en dicha subgráfica (véase [Conjuntos de Reglas](#)). Finalmente en la sección

[RepositorySetup](#) se genera una colección para la organización del sitio cuyos miembros son de la clase `swv:Person` y deben describirse con base en lo establecido en el `RuleSet` que se acaba de definir.

Elementos

abbreviation

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece el prefijo para los URI de la ontología generada y una abreviación para referencias a las reglas de este archivo de definiciones.	Atributo de Realm .

Aparece en	Cardinalidad y orden
Realm	1..1, sin restricciones de orden.

El argumento es la abreviación de la ontología y referencia para las reglas que define el archivo de definiciones.

Ejemplo:

En la sección [Realm](#) se establece la abreviación:

```
<Realm>
  (...)
  abbreviation swv
  (...)
</Realm>
```

Entonces los URI de los recursos de la ontología generada deben llevar este prefijo:

```
<Property swv:hasName>
  (...)
</Property>
```

Asimismo, se emplea la misma abreviación para referir este archivo de definiciones desde otro archivo de definiciones:

```
<KRRelation has_accessible_digital_copy>
  swv.content.has_system_accessible_digital_copy
</KRRelation>
```

Véase también [Referencia a una KRRelationRule](#) y [dependsRealm](#).

AbstractDataType

Tipo	Identificador de sección	Resumen	Relación con el código
sección	URI de un recurso RDE	Genera un tipo de datos abstracto .	Genera un objeto AbstractDataType .

Aparece en	Cardinalidad y orden
Vocabulary	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	1..1, y debe ser el primer elemento de la sección.
Documentación incrustada multilingüe	0..1, y si aparece debe ser el segundo elemento de la sección.
subDataTypeOf	0..*, sin restricciones de orden.
comparisonVector	0..*, sin restricciones de orden.

Ejemplo:

```
<AbstractDataType swv:CalendarPointer>
  "Calendar Pointer"@en
  "Indicador sobre el calendario"@es
  subDataTypeOf swv:TimeData
</AbstractDataType>
```

anchor

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece un fragmento de descripción como el objetivo de un enlace interno de una descripción .	Atributo de DescFragmentTemplate .

Aparece en	Cardinalidad y orden
UniqueDF	0..1, sin restricciones de orden.

Normalmente el argumento de la directriz es un [identificador de objeto](#). Se puede generar un vínculo hacia este objetivo por medio de una sección [linkTo](#) con el mismo identificador de objeto.

El argumento también puede ser la cadena **VALUE**; en este caso se establece el fragmento de descripción como objetivo de cualquier vínculo que aparezca en la misma descripción y que señale el mismo objeto describable que menicone este fragmento de descripción.

Area

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Define una área al interior de una descripción .	Genera un objeto DescAreaTemplate , el cual se usa para generar un DescArea en el lugar apropiado en una Description .

Aparece en	Cardinalidad y orden
FullDesc	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
------------	----------------------

Title	Después de la ejecución del código enrustado : 1..1, y debe resultar el primer elemento de la sección.
Block	0..*, sin restricciones de orden.
df	0..*, sin restricciones de orden.
UniqueDF	0..*, sin restricciones de orden.
textBPV	0..*, sin restricciones de orden.
imageBPV	0..*, sin restricciones de orden.
uniqueTextBPV	0..*, sin restricciones de orden.
uniqueImageBPV	0..*, sin restricciones de orden.
Texto multilingüe	0..*, sin restricciones de orden.
Código enrustado en plantillas de descripciones	0..*, sin restricciones de orden.

Ejemplo:

Véase [Tipos y configuración de descripciones](#).

bindSOCToRuleSet

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Vincula la descripción de una colección para la organización del sitio con un RuleSet determinado; se usa para configurar la descripción archivística a nivel de colección.	El objeto DescribableObject (del cual SOC es descendiente) tiene un atributo que registra su vínculo con un RuleSet .

Aparece en	Cardinalidad y orden
SOC	0..1, sin restricciones de orden.

El argumento debe ser un [identificador de objeto](#) de un [RuleSet](#) definido por este archivo de definiciones u otro del cual este archivo depende (véase [dependsRealm](#)).

Por el momento éste es el único mecanismo que ofrece el sistema para vincular un individuo específico con un [RuleSet](#); a futuro se diseñarán maneras de establecer vínculos de este tipo de otros contextos.

Ejemplo:

```
<SOC photographic_prints>
  "Juan Pérez photographic positives fond"@en
  "Fondo de positivos fotográficos Juan Pérez"@es
  hasGroupDomain swv:PhotographicPrint
  bindToRuleSet photographic_print
  mainLevel
  bindSOCToRuleSet photographic_print_soc
</SOC>
```

Aquí se establece que la descripción global de la colección para la organización del sitio `photographic_prints` obedece a las reglas que define el `RuleSet photographic_prints_soc`.

bindToRuleSet

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Vincula un los valores de un tipos de datos determinado o los miembros de una colección para la organización del sitio determinada con un RuleSet determinado.	Los objetos FundamentalDataType , ComplexDataType y SOC tienen atributos que registran su vínculo con un RuleSet .

Aparece en	Cardinalidad y orden
FundamentalDataType	0..1, sin restricciones de orden.
ComplexDataType	0..1, sin restricciones de orden.
SOC	1..1, sin restricciones de orden.

El argumento debe ser el [identificador de objeto](#) de un [RuleSet](#) definido por este archivo de definiciones u otro del cual este archivo depende (véase [dependsRealm](#)).

Ejemplos:

Aquí se vinculan los valores del tipo `swv:TimeRangeWSE` con el `RuleSet` `time_range_wse`:

```
<ComplexDataType swv:TimeRangeWSE>
  "Time Range with Specified Start and End Points"@en
  "Rango de tiempo con puntos inicial y final definidos"@es
  comment "A range of time with specified start and end points"@en
  comment "Un rango de tiempo con los puntos inicial y final
definidos"@es
  subDataTypeOf swv:TimeRange
  bindToRuleSet time_range_wse
</ComplexDataType>
```

En este ejemplo se vinculan los miembros de la colección para la organización del sitio `people` con el `RuleSet` `person`:

```
<SOC people>
  "People mentioned in collecitons in these archives"@en
  "Personas mencionadas en las colecciones de estos archivos"@es
  hasGroupDomain swv:Person
  bindToRuleSet person
  secondaryLevel
</SOC>
```

Block

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Define un bloque al interior de una Descripción .	Genera un objeto DescBlockTemplate .
Aparece en	Cardinalidad y orden		

FullDesc	0..*, sin restricciones de orden.
ShortDesc	0..*, sin restricciones de orden.
Area	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Title	Después de la ejecución del código enrustado : 0..1, y si aparece, debe resultar el primer elemento de la sección.
df	0..*, sin restricciones de orden.
UniqueDF	0..*, sin restricciones de orden.
Código enrustado en plantillas de descripciones	0..*, sin restricciones de orden.

A diferencia de una [Area](#), un **Block**, después de la ejecución del código enrustado, sólo puede contener un título y fragmentos de descripción; no se permite la inclusión directa de los valores de base para la presentación ni el [Texto multilingüe](#).

Ejemplo:

Véase [Tipos y configuración de descripciones](#).

Bloques de código Ruby

Tipo	Resumen	Relación con el código
bloque de código	Bloque de código ejecutable en el lenguaje Ruby.	Genera un objeto RubyCodeBlock .

Aparece en	Cardinalidad y orden
TextBPVFunction	0..1, sin restricciones de orden.
ImageBPVFunction	0..1, sin restricciones de orden.
InferenceRule	1..1, sin restricciones de orden.
CompVectorFunction	1..1, sin restricciones de orden.
Conditions	1..*, sin restricciones de orden.
Como argumento de uniqueTextBPV	n/a
Como argumento de uniqueImageBPV	n/a
Como argumento de label	n/a

El código Ruby se delimita con llaves (**{** y **}**). Cuando aparece un bloque de código en una [TextBPVFunction](#) o una [ImageBPVFunction](#), o como argumento de [uniqueTextBPV](#), [uniqueImageBPV](#) o [label](#) entonces devuelve un objeto BPV. Cuando aparece en una [CompVectorFunction](#) devuelve un valor que debe ser un objeto del tipo definido por el [comparisonVector](#) correspondiente. Y un bloque de código en una sección [Conditions](#) debe devolver un valor booleano.

Acerca de los objetos especiales disponibles en los bloques de código, véase [Referencia a una KRRelationRule](#), [Referencia a un RDFNode](#), [Solicitudes de relaciones de RC concretas](#) y [Referencia a un BPV](#).

Nótese que el método especial **assign** de las [referencias a una KRRelationRule](#) está disponible sólo en un bloque de código que aparece en una [InferenceRule](#).

BPVFunctions

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Sección de un RuleSet para las funciones de valores de base para la presentación .	El objeto RuleSet almacena un listado de los BPVFunction que incluye.
Aparece en		Cardinalidad y orden	
RuleSet		0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha sección.	
Contenidos			Cardinalidad y orden
Funciones de valores de base para la presentación (es decir, secciones ImageBPVFunction o TextBPVFunction)			0..*, sin restricciones de orden.

Class

Tipo	Identificador de sección	Resumen	Relación con el código
sección	URI de un recurso RDF	Define una clase que se incluirá en la ontología generada por este archivo de definiciones.	Genera un objeto Class .
Aparece en		Cardinalidad y orden	
Vocabulary		0..*, sin restricciones de orden.	
Contenidos		Cardinalidad y orden	
Texto multilingüe		1..1, y debe ser el primer elemento de la sección.	
Documentación incrustada multilingüe		0..1, y si aparece debe ser el segundo elemento de la sección.	
subClassOf		0..1, sin restricciones de orden.	

El [texto multilingüe](#) define el nombre amigable de la clase.

Código encrustado en plantillas de descripciones

Tipo	Resumen	Relación con el código
bloque de código	Bloque de código ejecutable en el lenguaje Ruby. Controla el flujo de ejecución o genera dinámicamente elementos de un RootDescTemplate .	Genera un objeto CodeInDescTemplate .
Aparece en		Cardinalidad y orden
FullDesc , o incrustado en cualquier elemento contenido en una FullDesc .		0..*, sin restricciones de orden.
ShortDesc , o incrustado en cualquier elemento contenido en una ShortDesc .		0..*, sin restricciones de orden.

Existen dos tipos de código encrustado en las plantillas:

Código que inserta una cadena en la plantilla

Se inserta en la plantilla como cadena el valor que devuelve el código. Se delimita el bloque de código con un `<%=` y un `%>`.

Código que no inserta una cadena en la plantilla

Típicamente se usa para el control de flujo en la plantilla. Se delimita el bloque de código con un `<%` y un `%>`.

(A futuro se definirán algunas limitaciones acerca de cómo y donde el código puede incrustarse en las plantillas.)

Véase también [Plantilla de descripción fija](#).

comment

Véase [Documentación incrustada multilingüe](#).

comparisonVector

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	2 ó 3	Establece un mecanismo de comparación entre los valores de uno o más tipos de datos.	Genera un objeto ComparisonVector .
Aparece en		Cardinalidad y orden	
AbstractDataType		0..*, sin restricciones de orden.	
FundamentalDataType		0..*, sin restricciones de orden.	
ComplexDataType		0..*, sin restricciones de orden.	

Esta directriz permite establecer un mecanismo de comparación entre valores. (Véase [Los vectores de comparación](#).)

Para cada `comparisonVector` que se defina para un tipo de datos determinado, deben existir [CompVectorFunction](#) vinculadas con todos los valores de dicho tipo de datos, incluyendo los valores de los sub-tipos de datos. (Los [CompVectorFunction](#) se establecen en los [RuleSet](#).)

El primer argumento de la directriz `comparisonVector` es el nombre de una clase Ruby que puede emplearse en la comparación de valores (por ejemplo `Integer`). Los [bloques de código Ruby](#) en las [CompVectorFunction](#) deben devolver un objeto de la clase señalada.

El segundo argumento de la directriz `comparisonVector` es un [identificador de objeto](#) que se asigna a este vector de comparación. Las [CompVectorFunction](#) usan el mismo identificador para indicar vincularse con este `comparisonVector`.

El tercer argumento, si aparece, debe ser la cadena `default`. Señala que este `comparisonVector` constituye la manera predeterminada de compara los valores del tipo de datos establecida por la sección donde aparece la directriz.

Es posible que a futuro se modifique este mecanismo en función de las necesidades que surjan en los escenarios de uso concretos.

Ejemplo:

Se establece el tipo de datos abstracto `swv:CalendarPointer`, cuyos descendientes son tipos de

datos que señalan un punto o unidad en el calendario. Para este tipo de datos abstracto se define el vector de comparación **point**:

```
<AbstractDataType swv:CalendarPointer>
  "Calendar Pointer"@en
  "Indicador sobre el calendario"@es
  subDataTypeOf swv:TimeData
  comparisonVector Integer point default
</AbstractDataType>
```

Los tipos de datos descendientes de **swv:CalendarPointer** proporcionan diversos mecanismos para señalar puntos en el calendario; se puede mencionar sólo el año, el año y el mes, o el año, el mes, y el día. El tipo de datos fundamental **swv:YearInCommonEra** implementa la primera de estas posibilidades: permite indicar un punto en el calendario señalando únicamente el año. (Nota: por ahora en los datos de ejemplo, no aparecen los demás descendientes de **swv:CalendarPointer**.)

```
<FundamentalDataType swv:YearInCommonEra>
  "Year in the Common Era"@en
  "Año en la era común"@es
  subDataTypeOf swv:CalendarPointer
  bindToRuleSet year_in_common_era
</FundamentalDataType>
```

Este tipo de datos se encuentra vinculado al conjunto de reglas **year_in_common_era**, en el cual se establece una **CompVectorFunction** correspondiente al **comparisonVector** establecido por el super-tipo:

```
<RuleSet year_in_common_era>
  <CompVectorFunctions>
    <CompVectorFunction point>
      { Time.utc(LIT.textBPV).to_i }
    </CompVectorFunction>
  </CompVectorFunctions>
</RuleSet>
```

El bloque de código Ruby simplemente convierte el año a un número entero que puede emplearse en las comparaciones entre los valores asignados a cualquiera de los descendientes de **swv:CalendarPointer**.

ComplexDataType

Tipo	Identificador de sección	Resumen	Relación con el código
sección	URI de un recurso RDF	Genera un tipo de datos complejo .	Genera un objeto ComplexDataType
Aparece en	Cardinalidad y orden		
Vocabulary	0..*, sin restricciones de orden.		
Contenidos		Cardinalidad y orden	
Texto multilingüe		1..1, y debe ser el primer elemento de la sección.	
Documentación incrustada multilingüe		0..1, y si aparece debe ser el segundo elemento de la sección.	
subDataTypeOf		0..*, sin restricciones de orden.	
comparisonVector		0..*, sin restricciones de orden.	

[bindToRuleSet](#)

0..1, sin restricciones de orden.

CompVectorFunction

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Establece una función vinculada a un vector de comparación.	Genera un objeto ComparisonVector .
Aparece en		Cardinalidad y orden	
CompVectorFunctions		1..*, sin restricciones de orden.	
Contenidos		Cardinalidad y orden	
Bloque de código Ruby		1..1, sin restricciones de orden.	

El identificador de la sección debe ser el [identificador de objeto](#) de un [comparisonVector](#) válido para este tipo de datos (o uno de sus ancestros en la jerarquía de tipos de datos). El bloque de código Ruby debe devolver un objeto de la clase establecida por el [comparisonVector](#) referido.

CompVectorFunctions

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Agrupar las CompVectorFunction que se incluyen en un RuleSet determinado.	Un atributo de RuleSet almacena todas las CompVectorFunction correspondientes.
Aparece en		Cardinalidad y orden	
RuleSet		0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha sección.	
Contenidos		Cardinalidad y orden	
CompVectorFunction		1..*, sin restricciones de orden.	

Conditions

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Agrupar las condiciones adicionales que se imponen para que una regla de relación de RC pueda generar una relación de RC .	Un atributo de KRRelationRule registra las KRRelationRuleCondition vinculadas con la regla.
Aparece en		Cardinalidad y orden	
KRRelationRule		0..1, y si aparece, debe aparecer después de <code>defaultOrderBy</code> o, si éste no	

aparece, después de la referencia a una `KRRelationRule`.

Contenidos	Cardinalidad y orden
Bloque de código Ruby	1..*, sin restricciones de orden.

Los bloques de código Ruby deben devolver un valor booleano.

Nótese que las `Conditions` *no dicen nada* acerca de cómo debe estructurarse la gráfica RDF; únicamente determinan cuándo las `KRRelationRule` generan un vínculo entre dos nodos de una estructura existente. En cambio las [reglas de estructura](#) *sí* determinan cómo debe estructurarse dicha gráfica.

copyOfTemplate

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Aparece en una sección que genera una plantilla de fragmento de descripción; señala que dicha plantilla es una copia de una plantilla existente.	Atributo de DFTemplateConfig .

Aparece en	Cardinalidad y orden
CreateTemplate	0..1, sin restricciones de orden.
UniqueDF	0..1, sin restricciones de orden.

Esta directriz hace que la plantilla de fragmento de descripción generada por la sección donde aparezca sea inicialmente una copia de otra plantilla. En tanto, las configuraciones que se establezcan en dicha sección sólo modifican las configuraicones ya establecidas en la plantilla copiada.

El argumento debe ser el [identificador de objeto](#) de la plantilla que se quiere copiar. (Véase [RuleSet](#).)

CreateTemplate

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Genera una plantilla de fragmento de descripción nueva, que puede ser aprovechada en la configuración de descripciones de diversos tipos.	Genera un DFTemplateConfig el cual genera una DescFragmentTemplate , que es registrado en un atributo de RuleSet .

Aparece en	Cardinalidad y orden
NonDefaultDescFragments	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
copyOfTemplate	0..1, y si aparece, debe ser el primer elemento de la sección.
Texto multilingüe	0..1, y si aparece, debe ser ubicarse después de la directriz <code>copyOfTemplate</code> o, si ésta no aparece, al principio de la sección.

label	0..1, 0..1, y si aparece, debe ser ubicarse después de la directriz <code>copyOfTemplate</code> o, si ésta no aparece, al principio de la sección. (No pueden existir en la misma sección <code>UniqueDF</code> un elemento label y otro de texto multilingüe .)
imageBPV	0..1, sin restricciones de orden.
textBPV	0..1, sin restricciones de orden.
uniqueImageBPV	0..1, sin restricciones de orden.
uniqueTextBPV	0..1, sin restricciones de orden.

El identificador de la sección se emplea posteriormente en la configuración de las descripciones para referir la plantilla de fragmento de descripción generada por esta sección.

defaultImageBPVFunction

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece el método predeterminado de generar una representación en formato de imagen de un nodo vinculado al RuleSet donde aparece esta directriz.	Un atributo de los objetos RuleSet registra la función señalada por esta directriz.
Aparece en		Cardinalidad y orden	
ImageBPVFunction		0..1, y si aparece debe ser el primer elemento.	

Esta directriz se emplea sin argumentos. Señala que el bloque de código que aparece en la misma sección [ImageBPVFunction](#) constituye la función predeterminada para generar una representación en formato de imagen de un nodo vinculado con un [RuleSet](#).

Si no aparece la directriz en ninguna sección [ImageBPVFunction](#) de un [RuleSet](#)--y si incluso el [RuleSet](#) no establece [ImageBPVFunction](#) alguna--entonces el sistema crea automáticamente una función para generar una representación en formato de imagen.

Si existen varias secciones [ImageBPVFunction](#) de un [RuleSet](#) determinado, la directriz sólo puede aparecer en una de ellas.

defaultOrderBy

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Señala la manera predeterminada de ordenar un conjunto de nodos vinculados al RuleSet donde aparece la directriz.	Un atributo de RuleSet registra la información que señala esta directriz.
Aparece en		Cardinalidad y orden	
Descriptor		0..1, sin restricciones de orden.	
KRRelationRule		0..1, y si aparece debe ser el segundo elemento.	

En la [generación de conjuntos](#) normalmente se ordenan los elementos del conjunto. Esta directriz permite especificar la manera predeterminada de hacerlo.

Si la directriz aparece en una sección [Descriptor](#), entonces de manera predeterminada se ordena el conjunto con base en los nodos de salida que arroje la [regla de relación de RC](#) generada por dicha sección. (Véase [Descriptor](#) para una explicación acerca de las reglas generadas por esta sección.)

De la misma manera, si la directriz aparece en una sección [KRRelationRule](#), entonces de manera predeterminada se ordena el conjunto con base en los nodos de salida que arroje la regla especificada.

En ambos casos es sencillo el algoritmo para ordenar con base en los nodos de salida: Si los nodos son valores con un mismo [vector de comparación](#) predeterminado, se usa dicho vector. En cualquier otro caso, se ordenan los nodos con base en sus [valores de base para la presentación de texto](#) predeterminados.

Esta directriz no puede aparecer más de una vez en un RuleSet determinado. Si la directriz no aparece, se usa la regla generada por la primera sección [Descriptor](#) del RuleSet.

defaultTextBPVFunction

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece el método predeterminado de generar una representación en formato de texto de un nodo vinculado al RuleSet donde aparece esta directriz.	Un atributo de RuleSet registra la función señalada por esta directriz.
Aparece en		Cardinalidad y orden	
TextBPVFunction		0..1, y si aparece debe ser el primer elemento.	

Esta directriz se emplea sin argumentos. Señala que el bloque de código que aparece en la misma sección [TextBPVFunction](#) constituye la función predeterminada para generar una representación en formato de texto de un nodo vinculado con un [RuleSet](#).

Si no aparece la directriz en ninguna sección [TextBPVFunction](#) de un RuleSet--y si incluso el RuleSet no establece [TextBPVFunction](#) alguna--entonces el sistema crea automáticamente una función para generar una representación en formato de texto.

Si existen varias secciones [TextBPVFunction](#) de un RuleSet determinado, la directriz sólo puede aparecer en una de ellas.

Ejemplo:

De la RuleSet `person` en los datos de ejemplo:

```
<TextBPVFunction full_name_lf>
  defaultTextBPVFunction
  { (R00T->has_last_names).textBPV + ", " + (R00T-
>has_first_names).textBPV }
</TextBPVFunction>
```

DefinitionSet

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Permite agrupar las definiciones en bloques lógicos.	Genera un objeto DefinitionSet .

Aparece en	Cardinalidad y orden
(Elemento de nivel raíz en los archivos de definiciones.)	1..*, y debe aparecer después de la sección Realm .

Contenidos	Cardinalidad y orden
Vocabulary	0..*, sin restricciones de orden.
RuleSet	0..*, sin restricciones de orden.
RepositorySetup	0..*, sin restricciones de orden.

Nótese que los **DefinitionSet** no impactan directamente en la configuración de la capa de lógica de RC. Su finalidad es esencialmente facilitar la organización y lectura (por seres humanos) de los archivos de definiciones, dividiendo éstos en secciones lógicas.

A diferencia de lo que quizá se esperaría, las referencias a los **DefinitionSet** *no* forman parte de las [referencias a una KRRelationRule](#).

dependsRealm

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece una dependencia entre un archivo de definiciones y otro.	Atributo de Realm .

Aparece en	Cardinalidad y orden
Realm	0..*, sin restricciones de orden.

El argumento es la abreviación de otro archivo de definiciones (establecida por medio de la directriz [abbreviation](#) en aquel archivo).

Para poder referir las reglas de otro archivo de definiciones, es necesario mencionarlo por medio de esta directriz. (Véase [Referencia a una KRRelationRule](#).)

Para poder emplear los términos de una ontología generada por otro archivo de definiciones, también es necesario mencionarlo por medio de esta directriz. (Véase [URI de los recursos RDF](#).)

Además, cuando se refiere otro archivo de definiciones con esta directriz, se vuelven disponibles los vocablos de las ontologías externas que importa aquel archivo por medio de la directriz [externalOntology](#).

Las dependencias establecidas por la directriz no se encadenan. Es decir, si archivo *A* depende de archivo *B*, y archivo *B* depende de archivo *C*, esto *no* implica automáticamente que archivo *C* dependa de archivo *A*; en ese caso, para que exista una dependencia entre *C* y *A*, ésta tendrá que establecerse explícitamente en *C*.

Ejemplo:

En el archivo de definiciones [archivo-juan-perez.def](#), en la sección [Realm](#), se establece la siguiente dependencia:

```
dependsRealm swv
```

Esto permite referir, desde ese archivo, las reglas de [shared-worldview.def](#), tal como se hace en el RuleSet `photo_collection`:

```
<Descriptor has_main_content_link>
  useDefinition swv.manifestation.has_main_content_link
</Descriptor>
```

También hace posible referir los vocablos de la ontología generada por `shared-worldview.def`:

```
<Descriptor has_unique_id>
  property swv:hasUniqueIdentifier
  minCardinality 1
  maxCardinality 1
  outRequirement fromPropRange
</Descriptor>
```

Descriptions

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Sección que agrupa las configuraciones de las descripciones de un RuleSet .	Genera un objeto Descriptions .
Aparece en	Cardinalidad y orden		
RuleSet	0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha sección.		
Contenidos		Cardinalidad y orden	
NonDefaultDescFragments		0..1. Si aparece, debe ser el primer elemento.	
VariableDesc		1..1. Debe aparecer al principio de la sección, o después de NonDefaultDescFragments , si aparece éste.	
ShortDesc		1..1. Debe aparecer después de VariableDesc .	
FullDesc		1..1. Debe aparecer después de ShortDesc .	

Descriptor

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Establece una regla de estructura de base de la gráfica RDF, así como una regla de relación de RC y una plantilla de fragmento de descripción correspondientes.	Genera un objeto Descriptor , a partir del cual se crean una BaseStructureRule , una SingleDescriptorRelRule y una DescFragmentTemplate .
Aparece en	Cardinalidad y orden		
Structure	1..*, sin restricciones de orden.		
Contenidos		Cardinalidad y orden	
property		1..1, sin restricciones de orden.	
minCardinality		0..1, sin restricciones de orden.	
maxCardinality		0..1, sin restricciones de orden.	
maxNonInfCardinality		0..1, sin restricciones de orden.	

multilingual	0..1, sin restricciones de orden.
multipleOrdered	0..1, sin restricciones de orden.
outClass	0..1, sin restricciones de orden.
outDataType	0..1, sin restricciones de orden.
outFromGroup	0..1, sin restricciones de orden.
outRequirement	0..1, sin restricciones de orden.
outValue	0..1, sin restricciones de orden.
useDefinition	0..1, sin restricciones de orden.
defaultOrderBy	0..1, sin restricciones de orden.

Establece que un nodo vinculado con el [RuleSet](#) puede o debe ser (según lo establecido por [minCardinality](#) y [maxCardinality](#)) el sujeto de un [descriptor](#) con el predicado que señale la directriz [property](#) y un objeto que cumpla con lo establecido por [outRequirement](#). Todo lo anterior queda plasmado en una regla de estructura de base.

Asimismo, se generan una regla de relación de RC y una plantilla de fragmento de descripción correspondientes. Dichos objetos pueden ser referidos desde otros directrices por medio del [identificador de objeto](#) de la sección.

df

df es un elemento polivalente cuyo significado cambia según dónde aparece. A pesar de esta variación, se espera que para los usuarios el identificar su significado en cada contexto resulte una tarea sencilla e intuitiva.

df siempre remite de una manera u otra a un [fragmento de descripción](#).

df en las [plantillas de descripción fija](#)

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Inserta un fragmento de descripción en una descripción .	A la hora de procesar un RootDescTemplate , y con base en un DescFragmentTemplate existente, inserta un DescFragment en el lugar apropiado en una Description .
Aparece en	Cardinalidad y orden		
ShortDesc	0..*, sin restricciones de orden.		
Area	0..*, sin restricciones de orden.		
Block	0..*, sin restricciones de orden.		

El argumento debe ser el [identificador de objeto](#) de una plantilla de fragmento de descripción generada por otra sección del mismo [RuleSet](#).

A continuación, las posibles maneras concretas en que puede generarse la plantilla de fragmento de

descripción referida:

1. Desde una sección [Descriptor](#). Las secciones [Descriptor](#) siempre crean automáticamente una regla de relación de RC y una plantilla de fragmento de descripción. En este caso el argumento de la directriz `df` es el mismo identificador de objeto de la sección [Descriptor](#). (Véase [RuleSet](#) y [Descriptor](#) para una explicación acerca de la generación automática de objetos en este contexto.)
2. Desde una sección [KRRelationRule](#). Las secciones [KRRelationRule](#) siempre crean automáticamente una plantilla de fragmento de descripción. En este caso el argumento de la directriz es el mismo identificador de objeto de la sección [KRRelationRule](#). (Véase [RuleSet](#) y [KRRelationRule](#).)
3. Desde una sección [CreateTemplate](#) ubicada en una sección [NonDefaultDescFragments](#). En este caso el argumento de la directriz `df` debe ser el identificador de la sección [CreateTemplate](#) que aparece en la sección [NonDefaultDescFragments](#) y que genera la plantilla de fragmento de descripción.

Ejemplo:

En el archivo de definiciones [archivo-juan-perez.def](#), en el [RuleSet](#) `photographic_print`, aparece esta sección [Descriptor](#):

```
<Descriptor has_unique_id>
  property swv:hasUniqueIdentifier
  minCardinality 1
  maxCardinality 1
  outRequirement fromPropRange
</Descriptor>
```

También en el mismo [RuleSet](#) aparece esta sección [KRRelationRule](#):

```
<KRRelationRule has_title>
  has_main_content->image_as_content.has_title
</KRRelationRule>
```

Después, en la sección [Descriptions](#) se configura la descripción corta de la siguiente manera:

```
<ShortDesc>
  df has_unique_id
  df has_title
  (...)
</ShortDesc>
```

La primera directriz `df` remite a la plantilla de fragmento de descripción generada por la mencionada sección [Descriptor](#). La segunda, a la plantilla generada desde la mencionada sección [KRRelationRule](#). A continuación, una muestra de cómo *podría* verse el inicio de la descripción corta de un objeto descriptible específico vinculado a este [RuleSet](#):

Identificador único CMICH0421
Título "Puente de orizaba"

df en las secciones [VariableDesc](#)

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece la posibilidad de incluir un fragmento	Un atributo de VariableDescTemplateOpts registra las DescFragmentTemplate

	de descripción en una descripción variable, cuyos contenidos se eligen de manera dinámica.	disponibles para elaborar una VariableDescTemplate
--	--	--

Aparece en	Cardinalidad y orden
VariableDesc	0..*, sin restricciones de orden.

La función de los elementos **df** en una sección [VariableDesc](#) es similar a [la que desempeñan en las plantillas de descripción fija](#). La diferencia principal es que en lugar de insertar un fragmento de descripción en la descripción, los elementos **df** en este caso establecen *la posibilidad* de hacerlo, posibilidad que puede realizarse o no dependiendo de necesidades descriptivas que se determinen de manera dinámica. (Recuerda que la descripción variable es una descripción cuyos contenidos se eligen sobre el vuelo, por ejemplo, en los resultados de búsqueda. Véase [Tipos y configuración de descripciones](#).)

En este contexto, el argumento de la directriz **df** tiene la misma función y uso que cuando la directriz [aparece en las plantillas de descripción fija](#).

Ejemplo:

En la siguiente sección **VariableDesc** se mencionan diversos fragmentos de descripción **df** entre los cuales el sistema puede elegir para su inclusión en una descripción variable.

```
<VariableDesc>
  df has_unique_id
  df has_title
  df has_description
  df has_initial_photo_location
  df has_initial_photo_date
  df has_topic
  df has_notes
  df has_image_media_type
  df has_conservation_status
</VariableDesc>
```

Documentación incrustada multilingüe

Tipo	Formato de cada línea	Resumen	Relación con el código
elemento multi-línea	<code>comment</code> " <code><texto></code> "@ <code><idioma></code>	Permite incrustar en un archivo de definiciones comentarios en diversos idiomas que se incluyen en la ontología generada.	Genera un Comment .

Aparece en	Cardinalidad y orden
AbstractDataType	0..1, y si aparece debe ser el segundo elemento de la sección.
Class	0..1, y si aparece debe ser el segundo elemento de la sección.
ComplexDataType	0..1, y si aparece debe ser el segundo elemento de la sección.
FundamentalDataType	0..1, y si aparece debe ser el segundo elemento de la sección.
Property	0..1, y si aparece debe ser el segundo elemento de la sección.
Realm	0..1, y si aparece debe ser el segundo elemento de la sección.

RepositoryArea	0..1, y si aparece debe ser el segundo elemento de la sección.
SOC	0..1, y si aparece debe ser el segundo elemento de la sección.

En una sección determinada, siempre deben aparecer juntas las líneas que inician con **comment**. El conjunto de líneas se considera un sólo elemento. El formato de cada línea debe ser el siguiente (donde *<texto>* es el texto del comentario, e *<idioma>* es el código de dos letras de un idioma determinado):

```
comment "<texto>"@<idioma>
```

Ejemplo:

Aquí una sección [Property](#) donde el [texto multilingüe](#) establece el nombre amigable de una propiedad y la documentación encrustada aporta una explicación más detallada:

```
<Property swv:hasRelatedLocation>
  "Related Location"@en
  "Ubicación relacionada"@es
  comment "This property functions mainly as a superproperty of other
more precisely defined ones"@en
  comment "Esta propiedad funciona principalmente como superpropiedad,
cuyos descendientes se definen de manera más precisa"@es
  (...)
</Property>
```

externalOntology

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	2	Establece la posibilidad de referir una ontología externa desde el archivo de definiciones.	Genera una Ontology que se registra como atributo de Realm .

Aparece en	Cardinalidad y orden
Realm	0..*, sin restricciones de orden.

El primer argumento es la abreviación de la ontología externa y el segundo es una cadena entrecomillada con la URI de la misma.

FullDesc

Este elemento puede ser sección o directriz.

Como sección:

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Genera una plantilla de descripción fija para la descripción completa de los objetos descriptibles vinculadas con un RuleSet .	Genera un FullDescTemplate .

Contenidos	Cardinalidad y orden
Area	0..*, sin restricciones de orden.

Block	0..*, sin restricciones de orden.
df	0..*, sin restricciones de orden.
UniqueDF	0..*, sin restricciones de orden.
textBPV	0..*, sin restricciones de orden.
imageBPV	0..*, sin restricciones de orden.
uniqueTextBPV	0..*, sin restricciones de orden.
uniqueImageBPV	0..*, sin restricciones de orden.
Texto multilingüe	0..*, sin restricciones de orden.
Código encrustado en plantillas de descripciones	0..*, sin restricciones de orden.

Existen dos maneras de organizar los contenidos de una `FullDesc`:

1. Encerrar los elementos en una o más secciones [Area](#). En este caso no se pueden incluir directamente en la `FullDesc` los elementos [Block](#), [df](#), [UniqueDF](#), [textBPV](#), [imageBPV](#), [uniqueTextBPV](#), [uniqueImageBPV](#) o [Texto multilingüe](#), sino que éstos deben incluirse directa o indirectamente en una `Area`.
1. Sin usar secciones `Area`. En este caso los demás elementos pueden incluirse directamente en la `FullDesc`.

Como directriz:

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1; debe ser la cadena <code>default</code>	Señala que la descripción completa de los objetos descriptibles vinculados con un RuleSet debe elaborarse de manera predeterminada.	Se genera automáticamente un FullDescTemplate que implementa un algoritmo predeterminado.

Las reglas acerca de dónde puede aparecer este elemento no cambian si es sección o directriz:

Aparece en	Cardinalidad y orden
Descriptions	1..1, y debe aparecer después de ShortDesc .

FundamentalDataType

Tipo	Identificador de sección	Resumen	Relación con el código
sección	URI de un recurso RDF	Genera un tipo de datos fundamental .	Genera un objeto FundamentalDataType

Aparece en	Cardinalidad y orden
Vocabulary	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	1..1, y debe ser el primer elemento de la sección.
Documentación incrustada multilingüe	0..1, y si aparece debe ser el segundo elemento de la sección.
subDataTypeOf	0..*, sin restricciones de orden.

comparisonVector	0..*, sin restricciones de orden.
bindToRuleSet	0..1, sin restricciones de orden.

generalArchivalThings

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece que el área de repositorio generada por el archivo de definiciones es de uso estándar (es decir, no es para datos compartidos ni datos del sistema).	Atributo de RepositoryArea .

Aparece en	Cardinalidad y orden
RepositoryArea	0..1, sin restricciones de orden.

Para configurar un fondo o grupo de fondos específicos, normalmente se incluye esta directriz (en lugar de [sharedArchivalThings](#) o [systemThings](#), cuyo uso es más especializado) en la sección [RepositoryArea](#).

hasGroupDomain

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece la clase a la que deben pertenecer los miembros de una colección para la organización del sitio.	Atributo de PersistentGroup .

Aparece en	Cardinalidad y orden
SOC	1..1, sin restricciones de orden.

El argumento de la directriz debe ser el [URI](#) de la clase referida.

Identificador de objeto

Los identificadores de objetos son cadenas que se emplean para nombrar y referir objetos en los archivos de definiciones. Muchas directrices aceptan un identificador de objeto como argumento, y muchas secciones emplean un identificador de este tipo como identificador de sección.

Los identificadores de objeto pueden incluir letras, números, guiones y guiones bajos. Generalmente se sugiere usar letras minúsculas y separar las palabras con un guión bajo.

ImageBPVFunction

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Genera una función de valores de base para la presentación .	Genera un ImageBPVFunction .

Aparece en	Cardinalidad y orden
BPVFunctions	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
defaultImageBPVFunction	0..1, sin restricciones de orden.
Bloque de código Ruby	1..1, sin restricciones de orden.

El identificador de la sección permite referir la función generada en las [plantillas de descripción fija](#), por medio de la directriz [imageBPV](#).

imageBPV

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Inserta en una descripción, o establece como valor en una plantilla de fragmento de descripción, el resultado de una función de valor de base par la presentación de imagen .	Atributo de DFTemplateConfig y DescAreaTemplate .

Aparece en	Cardinalidad y orden
UniqueDF	0..1, sin restricciones de orden.
CreateTemplate	0..1, sin restricciones de orden.
ModifyTemplate	0..1, sin restricciones de orden.
FullDesc	0..*, sin restricciones de orden.
Area	1..1, sin restricciones de orden.
ShortDesc	1..1, sin restricciones de orden.

El argumento debe ser el identificador de objeto de una [ImageBPVFunction](#), [KRRelationRule](#) o [Descriptor](#).

InferenceRule

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Genera una regla de inferencia que aplica a los nodos vinculados con este RuleSet .	Genera una InferenceRule .

Aparece en	Cardinalidad y orden
InferenceRules	1..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Bloque de código Ruby	1..1, sin restricciones de orden.

Por ahora, las reglas de inferencia son simplemente [bloques de código Ruby](#) que generan estructuras RDF bajo determinadas condiciones. Es probable que a futuro se implementen otros mecanismos más estandarizados para generar inferencias.

En un bloque de código de este tipo se usa la palabra clave [ROOT](#) para referir el nodo vinculado con el

RuleSet. Asimismo, se emplea el método especial **assign** de las [referencia a una KRRelationRule](#) para generar las estructuras RDF.

Nótese que las reglas de inferencia de este tipo son limitados; sólo pueden partir de nodos vinculados con un **RuleSet**, por lo que no aplican necesariamente a todo el repositorio de conocimiento.

InferenceRules

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Sección que agrupa las reglas de inferencia de un RuleSet .	Genera un objeto InferenceRules .
Aparece en		Cardinalidad y orden	
RuleSet		0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha sección.	
Contenidos		Cardinalidad y orden	
InferenceRule		1..*, sin restricciones de orden.	

KRRelationRule

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Genera una regla de relación de RC así como una plantilla de fragmento de descripción correspondiente.	Genera una KRRelationRule y una DescFragmentTemplate .
Aparece en		Cardinalidad y orden	
KRRelationRules		1..*, sin restricciones de orden.	
Contenidos		Cardinalidad y orden	
Referencia a una KRRelationRule		1..1, y debe ser el primer elemento.	
defaultOrderBy		0..1, y si aparece, debe ser el segundo elemento.	
Conditions		0..1, y si aparece, debe aparecer después de defaultOrderBy o, si éste no aparece, después de la referencia a una KRRelationRule .	
StructRules		0..1, y si aparece, debe ser el último elemento.	

Al establecer la [regla de relación de RC](#), automáticamente se genera una plantilla de fragmento de descripción, que puede referirse por medio del mismo [identificador de objeto](#) de la sección.

KRRelationRules

Tipo	Identificador de sección	Resumen	Relación con el código
------	--------------------------	---------	------------------------

sección	ninguno	Sección que agrupa las reglas de relación de RC que establece un RuleSet .	Genera un objeto KRRelationRules .
Aparece en	Cardinalidad y orden		
RuleSet	0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha sección.		
Contenidos	Cardinalidad y orden		
KRRelationRule	1..*, sin restricciones de orden.		

label

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece la etiqueta de una plantilla de fragmento de descripción por medio de una función especializada.	Genera una ImageBPVFunction que se registra en un atributo de DFTemplateConfig y de la DescFragmentTemplate resultante.
Aparece en	Cardinalidad y orden		
UniqueDF	0..1, sin restricciones de orden.		
CreateTemplate	0..1, sin restricciones de orden.		
ModifyTemplate	0..1, sin restricciones de orden.		

El argumento debe ser un [bloque de código Ruby](#) que establece una [función de valor de base para la presentación de texto](#).

linkTo

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Configura una plantilla de fragmento de descripción para generar un vínculo desde un fragmento de descripción hacia otra parte de la misma descripción.	Atributo de DFTemplateConfig y de la DescFragmentTemplate resultante.
Aparece en	Cardinalidad y orden		
UniqueDF	0..1, sin restricciones de orden.		
Contenidos	Cardinalidad y orden		
Texto multilingüe	1..1, sin restricciones de orden.		

El identificador de la sección debe ser un [identificador de objeto](#) que remita un [anchor](#), un [Block](#) o una [Area](#).

LIT

Variable global disponible en los [bloques de código Ruby](#) de un [RuleSet](#) vinculado con un [FundamentalDataType](#). Representa el nodo literal de un valor al que se ha asignado dicho tipo de datos.

Ofrece el método `textBPV` que devuelve el [valor de presentación de base de texto](#) que se genere de una manera determinada a partir del nodo literal.

mainLevel

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece una colección para la organización del sitio como "de nivel principal".	Atributo de SOC .
Aparece en	Cardinalidad y orden		
SOC	0..1, sin restricciones de orden.		

maxCardinality

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece el número máximo de veces que puede aparecer un descriptor con el nodo vinculado a un RuleSet .	Atributo de StructureRule .
Aparece en	Cardinalidad y orden		
Descriptor	0..1, sin restricciones de orden.		
StructRules	0..1, sin restricciones de orden.		

El argumento debe ser un número entero.

maxNonInfCardinality

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece el número máximo de veces que puede aparecer un descriptor no generado por una regla de inferencia, con el nodo vinculado a un RuleSet .	Atributo de StructureRule .
Aparece en	Cardinalidad y orden		
Descriptor	0..1, sin restricciones de orden.		
StructRules	0..1, sin restricciones de orden.		

El argumento debe ser un número entero.

minCardinality

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece el número mínimo de veces que puede aparecer un descriptor con el nodo vinculado a un RuleSet .	Atributo de StructureRule .

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.
StructRules	0..1, sin restricciones de orden.

El argumento debe ser un número entero.

ModifyTemplate

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Modifica una plantilla de fragmento de descripción existente.	Genera un DFTemplateConfig el cual influye en la generación de una DescFragmentTemplate , que es registrado en un atributo de RuleSet

Aparece en	Cardinalidad y orden
NonDefaultDescFragments	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	0..1, y si aparece, debe ser ubicarse después de la directriz <code>copyOfTemplate</code> o, si ésta no aparece, al principio de la sección.
label	0..1, 0..1, y si aparece, debe ser ubicarse después de la directriz <code>copyOfTemplate</code> o, si ésta no aparece, al principio de la sección. (No pueden existir en la misma sección <code>UniqueDF</code> un elemento label y otro de texto multilingüe .)
imageBPV	0..1, sin restricciones de orden.
textBPV	0..1, sin restricciones de orden.
uniqueImageBPV	0..1, sin restricciones de orden.
uniqueTextBPV	0..1, sin restricciones de orden.

El identificador de la sección debe ser el [identificador de objeto](#) de la plantilla de fragmento de descripción que se pretende modificar. El mismo identificador se emplea posteriormente en la configuración de las descripciones para referir la plantilla de fragmento de descripción modificada por esta sección.

multilingual

Tipo	Número de argumentos	Resumen	Relación con el código
------	----------------------	---------	------------------------

directriz	0	Señala que un descriptor debe aceptar valores alternativos para diferentes idiomas.	Atributo de BaseStructureRule .
-----------	---	---	---

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.

multipleOrdered

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Señala que cuando el descriptor se presente varias veces con un nodo determinado vinculado con un RuleSet , dichos descriptors deben tener un orden preestablecido.	Atributo de BaseStructureRule .

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.

NonDefaultDescFragments

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Sección que agrupa las configuraciones especiales de las plantillas de fragmento de descripción de un RuleSet .	Genera un objeto NonDefaultDescFragments .

Aparece en	Cardinalidad y orden
Descriptions	0..1. Si aparece, debe ser el primer elemento.

Contenidos	Cardinalidad y orden
CreateTemplate	0..*, sin restricciones de orden.
ModifyTemplate	0..*, sin restricciones de orden.

orderUsing

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece una KRRelationRule que se emplea para determinar el orden de un conjunto de resultados de una TextBPVFunction .	Atributo de TextBPVFunction .

Aparece en	Cardinalidad y orden
TextBPVFunction	0..1, y si aparece, debe ubicarse después de la directriz

[defaultTextBPVFunction](#), o si no aparece ésta, debe ser el primer elemento.

Esta directriz es útil para las siguientes circunstancias: cuando el sistema genera un conjunto de objetos descriptibles, los ordena con base en "un campo" determinado, es decir, con base en los nodos de salida de una relación de RC determinada. Esto significa, en los hechos, ordenar dichos nodos de salida. Normalmente éstos se presentan al usuario mediante una [función de valores de base para la presentación](#); entonces es necesario ordenar los resultados de dichas funciones. Esta directriz señala una manera especial de ordenar dichos resultados, que se usa en lugar del mecanismo de ordenamiento estándar del idioma de la descripción.

El argumento debe ser una [referencia a una KRRelationRule](#) que anteriormente en el RuleSet se estableció como aplicable a los nodos vinculados a éste.

Ejemplo:

```
<TextBPVFunction full_name_fl>
  orderUsing has_last_names
  { (ROOT->has_given_names).textBPV + " " + (ROOT-
>has_last_names).textBPV }
</TextBPVFunction>
```

Esta función, que aparece en el RuleSet `person`, genera una representación textual de una persona colocando primero los nombres y después los apellidos. Sin embargo, la manera correcta de ordenar un listado de los valores de la función es con base en los apellidos.

OUT

Variable especial disponible únicamente desde un [bloque de código Ruby](#) que aparezca en una sección [Conditions](#). Devuelve un objeto [RDFNode](#) con el nodo que está siendo evaluado por el sistema para su posible integración como nodo de salida en una [relación de RC](#).

outClass

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que una relación de RC generada por una regla de relación de RC determinada debe tener como nodo de salida un objeto descriptible que pertenezca a la clase señalada.	Atributo de StructureRule .
Aparece en	Cardinalidad y orden		
Descriptor	0..1, sin restricciones de orden.		
StructRules	0..1, sin restricciones de orden.		

El argumento debe ser el [URI](#) de la clase referida.

outDataType

Tipo	Número de argumentos	Resumen	Relación con el código
------	----------------------	---------	------------------------

directriz	1	Establece que una relación de RC generada por una regla de relación de RC determinada debe tener como nodo de salida un valor del tipo de datos señalado.	Atributo de StructureRule .
-----------	---	---	---

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.
StructRules	0..1, sin restricciones de orden.

El argumento debe ser el [URI](#) del tipo de datos referido.

outFromGroup

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que una relación de RC generada por una regla de relación de RC determinada debe tener como nodo de salida un objeto describable que forme parte del grupo señalado.	Atributo de StructureRule .

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.
StructRules	0..1, sin restricciones de orden.

El argumento debe ser un [identificador de objeto](#) que refiere una colección para la organización del sitio establecida por una directriz [SOC](#).

outRequirement

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece un requisito sobre el nodo de salida de una relación de RC generada por una regla de relación de RC determinada.	Atributo de StructureRule .

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.
StructRules	0..1, sin restricciones de orden.

El único argumento permitido por el momento es la cadena `fromPropRange`, que señala que el requisito para el nodo de salida debe determinarse con base en el rango de la propiedad del último descriptor en la regla de relación de RC.

outValue

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que una relación de RC generada por	Atributo de

		una regla de relación de RC determinada debe tener como nodo de salida el nodo señalado.	StructureRule .
--	--	--	---------------------------------

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.
StructRules	0..1, sin restricciones de orden.

El argumento puede ser un [URI de un recurso RDF](#) o un [identificador de objeto](#). En el segundo caso, el identificador debe referir un elemento de un archivo de definiciones que genere un objeto descriptible (por ejemplo, [SOC](#)).

Plantilla de descripción fija

Una conjunto de directrices, secciones y bloques de código encrustado que establece cómo generar una [descripción](#). Existen dos elementos que generan plantillas de este tipo: [FullDesc](#) y [ShortDesc](#).

En teoría la plantilla requiere dos pasos de procesamiento: uno para ejecutar el código encrustado, y otro para procesar los demás elementos luego de las modificaciones a la plantilla efectuadas por medio el código encrustada.

Property

Tipo	Identificador de sección	Resumen	Relación con el código
sección	URI de un recurso RDF	Define una propiedad que se incluirá en la ontología generada por este archivo de definiciones.	Genera un objeto Property .

Aparece en	Cardinalidad y orden
Vocabulary	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	1..1, y debe ser el primer elemento de la sección.
Documentación incrustada multilingüe	0..1, y si aparece debe ser el segundo elemento de la sección.
subPropertyOf	0..1, sin restricciones de orden.
range	0..1, sin restricciones de orden.

El [texto multilingüe](#) define el nombre amigable de la propiedad, el cual se emplea de manera predeterminada en las etiquetas de las plantillas de fragmento de descripción.

Nótese la distinción que se hace entre una sección **Property** y la directriz (con letra minúscula) [property](#).

property

Tipo	Número de argumentos	Resumen	Relación con el código
------	----------------------	---------	------------------------

directriz	1	Señala la propiedad de un descriptor que puede o debe vincularse con un nodo ligado con un RuleSet .	Atributo de la SingleDescriptorRelRule ligada a esta sección Descriptor .
-----------	---	--	---

Aparece en	Cardinalidad y orden
Descriptor	1..1, sin restricciones de orden.

El argumento debe ser el [URI](#) de la propiedad referida.

Nótese la distinción que se hace entre una directriz **property** y la sección (con letra mayúscula) [Property](#).

range

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Señala el rango de una propiedad.	Atributo de Property .

Aparece en	Cardinalidad y orden
Property	0..1, sin restricciones de orden.

El argumento debe ser el [URI](#) del la clase o tipo de datos que constutuye rango de la propiedad.

Realm

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Establece los parámetros globales de un archivo de definiciones.	Genera un objeto Realm .

Aparece en	Cardinalidad y orden
(Elemento de nivel raíz en los archivos de definiciones.)	1..1, y debe aparecer al inico del archivo.

Contenidos	Cardinalidad y orden
Texto multilingüe	1..1, y debe ser el primer elemento de la sección.
Documentación incrustada multilingüe	0..1, y si aparece debe ser el segundo elemento de la sección.
abbreviation	1..1, sin restricciones de orden.
thisOntology	1..1, sin restricciones de orden.
dependsRealm	0..*, sin restricciones de orden.
externalOntology	0..*, sin restricciones de orden.
RepositoryArea	1..1, sin restricciones de orden.

Referencia a un BPV

En los [bloques de código Ruby](#) se emplean objetos que refieren los [valores de base para la presentación](#).

En las expresiones Ruby los valores de base para la presentación de texto pueden usarse de manera

intercambiable con los objetos `String`.

Por otra parte, los los valores de base para la presentación de imagen ofrecen los siguientes métodos:

`thumb`

Señala que la imagen debe presentarse en miniatura.

`fitViewPort`

Señala que la imagen debe reducirse de tamaño sólo si no cabe en el espacio disponible de la pantalla (u otro medio visual).

`full`

Señala que la imagen debe presentarse con la máxima resolución disponible.

Referencia a un `RDFNode`

En los [bloques de código Ruby](#) se emplean objetos que refieren directamente un nodo RDF. Dichos objetos pueden obtenerse de las siguientes maneras:

1. Por medio de las variables especiales [ROOT](#), [OUT](#) y [LIT](#).
2. Como resultado del método `node` sobre un [resultado de solicitud de relaciones de RC concretas](#).
3. Por medio del [URI de un recurso RDF](#).
4. Por medio del [identificador](#) de un elemento de un archivo de definiciones que genere un objeto descriptible.

Los métodos disponibles sobre los objetos `RDFNode` son:

`textBPV`

Devuelve el valor de base de presentación de texto predeterminado del nodo.

`imageBPV`

Devuelve el valor de base de presentación de imagen predeterminado del nodo.

`classes`

Realiza una solicitud de relaciones de RC concretas y devuelve un conjunto de resultados que contiene los nodos de las clases a las que pertenece el nodo referido.

Referencia a una `KRRelationRule`

Las referencias a las reglas de relación de RC aparecen en distintos puntos en los archivos de definiciones.

Sintaxis básico

Como se explicó en el apartado [Relaciones de RC](#), existen dos tipos de relación de RC: las `SingleDescriptorRelation` y las `MultipleDescriptorRelation`. Las relaciones del segundo tipo se construyen a partir del encadenamiento de varias relaciones del primer tipo.

Recordemos también que las reglas de relación de RC generan, precisamente, relaciones entre nodos, y que en dichas relaciones participan estructuras de la gráfica RDF generadas previamente.

Propiamente dicho, una regla de relación de RC *no* modifica la gráfica RDF; sólo establece un "atajo" para pasar de un nodo a otro en estructuras existentes. (Véase, más adelante, [Generación de estructuras desde el código](#) para una explicación de cómo las reglas de relación de RC pueden emplearse en la generación de subgráficas.)

Las reglas que crean una `SingleDescriptorRelation` típicamente se generan por medio de una sección [Descriptor](#). El "nombre" de la regla siempre es el [identificador de objeto](#) de dicha sección.

Por ejemplo en el `RuleSet` `person`, aparece una sección `Descriptor` con el identificador `has_last_names`:

```
<Descriptor has_last_names>
  property swv:hasLastNames
  minCardinality 1
  maxCardinality 1
  outRequirement fromPropRange
  defaultOrderBy
</Descriptor>image_as_content.has_title
```

Esta sección genera, entre otras cosas, una regla que crea unas `SingleDescriptorRelation`; dicha regla es referida más adelante en el mismo `RuleSet` por la directriz [orderUsing](#):

```
<TextBPVFunction full_name_fl>
  orderUsing has_last_names
  (...)
</TextBPVFunction>
```

A menudo es necesario referir un elemento proveniente de otro `RuleSet` o incluso de otro archivo de definiciones. En estas situaciones se anexa al principio del identificador el nombre del `RuleSet` y, en su caso, la abreviación del archivo de definiciones correspondiente. Por ejemplo, desde el archivo [archivo-juan-perez.def](#) se puede hablar de la regla de relación de RC

`has_main_content`--que aparece en el `RuleSet` `manifestation` en el archivo [shared-worldview.def](#)--de la manera siguiente:

```
swv.manifestation.has_main_content
```

Expresada formalmente, el sintaxis para referir una regla de relación de RC es:

```
[[abreviación del archivo de definiciones.]nombre del RuleSet.]nombre de la
relación de RC
```

Encadenamiento de reglas

Se usa el operador `->` para encadenar reglas establecidas desde una sección [Descriptor](#), desde una sección [KRRelationRule](#) o automáticamente por el mismo sistema. De esta manera se pueden referir rutas largas en la gráfica RDF. Por ejemplo, en [archivo-juan-perez.def](#), en el `RuleSet` `photographic_print` se encadenan dos reglas establecidas:

```
has_main_content->image_as_content.has_title
```

Técnicamente este código genera una nueva regla que crea unas `MultipleDescriptorRelation`. Funciona de la manera siguiente:

1. El nodo de entrada de una relación generada por la nueva regla es el nodo de entrada de una relación generada por la regla `has_main_content`.
2. El nodo de salida de la relación generada por `has_main_content` es el nodo de entrada de una relación generada por `image_as_content.has_title`.
3. El nodo de salida de la relación generada por la nueva regla es el nodo de salida de la relación generada por `has_main_content`.

Inversión de reglas

Se puede referir la inversión de una regla de relación de RC por medio del sufijo `.inv`.

Por ejemplo, aquí se crea una regla llamada `has_linked_manifestation` que es la inversión de la regla `manifestation.has_content_link`.

```
<KRRelationRule has_linked_manifestation>
  manifestation.has_content_link.inv
  (...)
</KRRelationRule>
```

Las reglas invertidas también pueden encadenarse:

```
<KRRelationRule has_system_accessible_digital_copy>
  content_link.has_linked_content.inv-
>content_link.has_linked_manifestation
  (...)
</KRRelationRule>
```

Relaciones no inferenciadas

A veces es necesario referir únicamente las relaciones de RC generadas por una regla sin pasar por estructuras de la gráfica creadas por medio de las [InferenceRule](#). Para ello, puede agregarse el sufijo `.nonInf` a la referencia para obtener una nueva regla derivada, que es igual a la regla referida, excepto que no puede pasar por estructuras de ese tipo.

Uso en bloques de código

En los [bloques de código Ruby](#) las referencias a una regla de relación de RC devuelvan objetos de tipo [KRRelationRule](#), los cuales pueden emplearse de distintas maneras. (Véase [Referencia a un RDFNode](#), [Solicitudes de relaciones de RC concretas](#) y [Referencia a un BPV](#).)

Generación de estructuras desde el código

Cuando un bloque de código Ruby aparece en una sección [InferenceRule](#) puede usarse, en dicho bloque, el método `assign` sobre el objeto [KRRelationRule](#). Al referir la regla se invocan automáticamente las [reglas de estructura](#) apropiadas que permiten generar una estructura nueva en la gráfica RDF. Éste es el único caso en que se emplean las reglas de relación RC para crear estructuras en el RC.

Reglas de estructura

Reglas que determinan la estructura de la gráfica RDF. Las directrices que pueden establecer reglas de estructura incluyen [minCardinality](#), [maxCardinality](#), [maxNonInfCardinality](#), [outClass](#), [outDataType](#), [outFromGroup](#), [outRequirement](#), [outValue](#), [multilingual](#) y [multipleOrdered](#).

Todas estas directrices pueden aparecer en [Descriptor](#), y la mayoría de ellas en [StructRules](#), también.

RepositoryArea

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Establece los parámetros globales del área de repositorio que genera el archivo de definiciones.	Configura un RepositoryArea .
Aparece en	Cardinalidad y orden		
Realm	1..1, sin restricciones de orden.		
Contenidos		Cardinalidad y orden	
Texto multilingüe		1..1, y debe ser el primer elemento de la sección.	
Documentación incrustada multilingüe		0..1, y si aparece debe ser el segundo elemento de la sección.	
generalArchivalThings		0..1, sin restricciones de orden.	
sharedArchivalThings		0..1, sin restricciones de orden.	
systemThings		0..1, sin restricciones de orden.	

RepositorySetup

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Agrupar las configuraciones específicas del área de repositorio correspondiente al archivo de definiciones.	Genera un objeto RepositorySetup
Aparece en	Cardinalidad y orden		
DefinitionSet	0..*, sin restricciones de orden.		
Contenidos	Cardinalidad y orden		
SOC	1..*, sin restricciones de orden.		

ROOT

Variable especial disponible únicamente desde un [bloque de código Ruby](#). Devuelve un objeto [RDFNode](#) con el nodo que el sistema está procesando en el momento de ejecución del código y que se encuentra vinculado al RuleSet en el cual aparece dicho código.

RuleSet

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Agrupar las configuraciones de operaciones de lógica de RC. (Véase Conjuntos de reglas .)	Genera un objeto RuleSet .
Aparece en	Cardinalidad y orden		

DefinitionSet	0..*, sin restricciones de orden.
-------------------------------	-----------------------------------

Contenidos	Cardinalidad y orden
Structure	0..1. Véase reglas de ordenamiento, a continuación.
InferenceRules	0..1. Véase reglas de ordenamiento, a continuación.
KRRelationRules	0..1. Véase reglas de ordenamiento, a continuación.
CompVectorFunctions	0..1. Véase reglas de ordenamiento, a continuación.
BPVFunctions	0..1. Véase reglas de ordenamiento, a continuación.
Descriptions	0..1. Véase reglas de ordenamiento, a continuación.

Los contenidos de un **RuleSet** deben ordenarse de la siguiente manera:

1. **Structure**
2. **InferenceRules**
3. **KRRelationRules**
4. **CompVectorFunctions**
5. **BPVFunctions**
6. **Descriptions**

Aún cuando se omiten uno o varios de los elementos, ningún elemento cambia de lugar en relación con los demás.

Los mecanismos de configuración de los **RuleSet** fueron diseñados para permitir la no repetición de información y la modularidad. La mayoría de las secciones generan objetos que pueden ser aprovechados desde otras secciones del mismo **RuleSet** e incluso desde otros **RuleSet**:

En la sección **Structure**, cada elemento [Descriptor](#) genera una regla de estructura de base, una regla de relación de RC y una plantilla de fragmento de descripción.

En la sección **KRRelationRules**, cada elemento [KRRelationRule](#) genera una regla de relación de RC y una plantilla de fragmento de descripción.

En la sección **CompVectorFunctions**, cada elemento [CompVectorFunction](#) genera una función vinculado con un vector de comparación.

En la sección **BPVFunctions**, cada elemento [ImageBPVFunction](#) y [TextBPVFunction](#) genera una función de valores de base para la presentación.

Y en la sección **Descriptions**, se generan plantillas de fragmento de descripción no estándares y se aprovechan los elementos generados desde otras secciones para configurar la elaboración de descripciones.

Como se mencionó en el apartado [Conjuntos de reglas](#), los **RuleSet** pueden vincularse con los valores de un determinado tipo de datos, con todos los miembros de una colección para la organización del sitio determinada, o con individuos específicos. Dicha vinculación se establece por medio de las directrices [bindToRuleSet](#) y [bindSOCToRuleSet](#).

Por el momento se propone que la vinculación de los valores de un tipo de datos con un **RuleSet** se herede de los supertipos de datos hacia sus descendientes, prohibiendo las jerarquías que impliquen una vinculación con más de un **RuleSet** de parte los valores de cualquier tipo de datos. Posiblemente a futuro cambien estas reglas de herencia.

secondaryLevel

Tipo	Número de	Resumen	Relación con el
------	-----------	---------	-----------------

	argumentos		código
directriz	0	Establece una colección para la organización del sitio como "de nivel secundario".	Atributo de SOC .

Aparece en	Cardinalidad y orden
SOC	0..1, sin restricciones de orden.

Típicamente las colecciones para la organización del sitio de nivel secundario no se despliegan ante los usuarios como colecciones normales; a menudo almacenan conjuntos de objetos descriptibles que se emplean en la descripción de los miembros de las colecciones de nivel principal. (Véase [mainLevel](#) y [repository.n3](#).)

sharedArchivalThings

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece que el área de repositorio generada por el archivo de definiciones es de uso compartido (es decir, no es estándar ni para datos del sistema).	Atributo de RepositoryArea .

Aparece en	Cardinalidad y orden
RepositoryArea	0..1, sin restricciones de orden.

Para configurar un fondo o grupo de fondos específicos, normalmente no se emplea esta directriz en el archivo de definiciones. (En cambio lo típico es usar la directriz [generalArchivalThings](#).)

ShortDesc

Este elemento puede ser sección o directriz.

Como sección:

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Genera una plantilla de descripción fija para la descripción breve de los objetos descriptibles vinculadas con un RuleSet .	Genera un ShortDescTemplate .

Contenidos	Cardinalidad y orden
Block	0..*, sin restricciones de orden.
df	0..*, sin restricciones de orden.
UniqueDF	0..*, sin restricciones de orden.
textBPV	0..*, sin restricciones de orden.
imageBPV	0..*, sin restricciones de orden.
uniqueTextBPV	0..*, sin restricciones de orden.
uniqueImageBPV	0..*, sin restricciones de orden.

Texto multilingüe	0..*, sin restricciones de orden.
Código encrustado en plantillas de descripciones	0..*, sin restricciones de orden.

Nótese que a diferencia de las [FullDesc](#), las [ShortDesc](#) no aceptan los elementos [Area](#).

Como directriz:

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1; debe ser la cadena default	Señala que la descripción breve de los objetos descriptibles vinculados con un RuleSet debe elaborarse de manera predeterminada.	Se genera automáticamente un ShortDescTemplate que implementa un algoritmo predeterminado.

Las reglas acerca de dónde puede aparecer este elemento no cambian si es sección o directriz:

Aparece en	Cardinalidad y orden
Descriptions	1..1, y debe aparecer después de ShortDesc .

SOC

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Establece una colección para la organización del sitio.	Genera un objeto SOC .

Aparece en	Cardinalidad y orden
RepositorySetup	1..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	1..1, y debe ser el primer elemento de la sección.
Documentación incrustada multilingüe	0..1, y si aparece debe ser el segundo elemento de la sección.
hasGroupDomain	1..1, sin restricciones de orden.
bindToRuleSet	1..1, sin restricciones de orden.
bindSOCToRuleSet	0..1, sin restricciones de orden.
mainLevel	0..1, sin restricciones de orden.
secondaryLevel	0..1, sin restricciones de orden.

Structure

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Agrupar los Descriptor de un RuleSet .	Genera un objeto Structure .

Aparece en	Cardinalidad y orden
RuleSet	0..1. Véase RuleSet para información acerca de las reglas de ordenamiento en dicha

sección.

Contenidos	Cardinalidad y orden
Descriptor	1..*, sin restricciones de orden.

StructRules

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Establece reglas de estructuración sobre una regla de relación de RC que no se encuentre plasmada en una sección Descriptor .	Genera un objeto IndirectStructureRule .

Aparece en	Cardinalidad y orden
KRRelationRule	0..1, y si aparece, debe ser el último elemento.

Contenidos	Cardinalidad y orden
minCardinality	0..1, sin restricciones de orden.
maxCardinality	0..1, sin restricciones de orden.
maxNonInfCardinality	0..1, sin restricciones de orden.
outClass	0..1, sin restricciones de orden.
outDataType	0..1, sin restricciones de orden.
outFromGroup	0..1, sin restricciones de orden.
outRequirement	0..1, sin restricciones de orden.
outValue	0..1, sin restricciones de orden.

Solicitudes de relaciones de RC concretas

En los [bloques de código Ruby](#) se puede conectar un objeto [RDFNode](#) con una regla de relación de RC para obtener las relaciones concretas generadas por dicha regla. Para ello, se usa el operador `->`. Una conexión de ese tipo se denomina una solicitud de relaciones de RC concretas.

En el siguiente ejemplo, se conecta una regla con el variable especial [ROOT](#) (que refiere automáticamente el nodo que el sistema está procesando en el momento de ejecución del código y que se encuentra vinculado al `RuleSet` en el cual aparece dicho código).

```
<TextBPVFunction text_ref>
  defaultTextBPVFuncrion
    { "\"" + (ROOT->has_title_text).textBPV + "\"" }
</TextBPVFunction>
```

El código `ROOT->has_title_text` devuelve un *resultado* de solicitud de relaciones concretas. Dicho resultado contiene la(s) relacion(es) generada(s) por `has_title_text` donde el nodo de entrada es `ROOT`.

En el bloque de código anterior se presupone (acertadamente) que el resultado de la solicitud contendrá una sola relación de RC. Al resultado se le aplica el método `textBPV`, que regresa el valor de base para la presentación de texto predeterminado del nodo de salida de primera relación devuelta.

Los métodos disponibles sobre un resultado de solicitud de relaciones concretas son:

hasObj

Regresa un valor booleano: verdadero si el resultado contiene uno o más nodos, falso si no contiene ninguno.

textBPV

Regresa el valor de base para la presentación de texto predeterminado del nodo de salida de la primera relación en el conjunto de resultados.

imageBPV

Regresa el valor de base para la presentación de imagen predeterminado del nodo de salida de la primera relación en el conjunto de resultados.

node

Regresa el objeto **RDFNode** del nodo de salida de la primera relación en el conjunto de resultados.

includes(*RDFNode*)

Regresa un valor booleano: verdadero si el conjunto de resultados incluye el nodo RDF, falso si no lo incluye.

Además, las funciones vinculadas con los [vectores de comparación](#) (creadas por medio de secciones [CompVectorFunction](#)) se encuentran disponibles como métodos sobre los resultados de solicitud de relaciones. De nuevo, en el caso de existir más de un nodo en los resultados, dichas funciones actúan sobre el primer nodo del conjunto.

Sin duda a futuro se establecerán métodos adicionales sobre los resultados de solicitudes de relaciones.

subClassOf

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que una clase es subclase de la clase señalada.	Atributo de Class .
Aparece en	Cardinalidad y orden		
Class	0..1, sin restricciones de orden.		

El argumento debe ser el [URI](#) de la clase que es la superclase de la clase definida por la sección donde aparece la directriz.

subDataTypeOf

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que un tipo de datos es subtipo de datos del tipo de datos señalado.	Atributo de DataType .
Aparece en	Cardinalidad y orden		
AbstractDataType	0..1, sin restricciones de orden.		
ComplexDataType	0..1, sin restricciones de orden.		
FundamentalDataType	0..1, sin restricciones de orden.		

El argumento debe ser el [URI](#) del tipo de datos que es el supertipo de datos del tipo de datos definido por la sección donde aparece la directriz.

subPropertyOf

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Establece que una propiedad es subpropiedad de la propiedad señalada.	Atributo de Property .
Aparece en	Cardinalidad y orden		
Property	0..1, sin restricciones de orden.		

El argumento debe ser el [URI](#) de la propiedad que es la superpropiedad de la propiedad definida por la sección donde aparece la directriz.

systemThings

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	0	Establece que el área de repositorio generada por el archivo de definiciones es para datos del sistema (es decir, no es estándar ni para datos compartidos).	Atributo de RepositoryArea .
Aparece en	Cardinalidad y orden		
RepositoryArea	0..1, sin restricciones de orden.		

Para configurar un fondo o grupo de fondos específicos, normalmente no se emplea esta directriz en el archivo de definiciones. (En cambio lo típico es usar la directriz [generalArchivalThings](#).)

En otras palabras, ¡no uses esta directriz! (por favor).

TextBPVFunction

Tipo	Identificador de sección	Resumen	Relación con el código
sección	Identificador de objeto	Genera una función de valores de base para la presentación de texto.	Genera un TextBPVFunction .
Aparece en	Cardinalidad y orden		
BPVFunctions	0..*, sin restricciones de orden.		
Contenidos		Cardinalidad y orden	
defaultTextBPVFunction		0..1, y si aparece debe ser el primer elemento.	
orderUsing		0..1, y si aparece, debe ubicarse después de la directriz defaultTextBPVFunction , o si no aparece ésta, debe ser el primer elemento.	

[Bloque de código Ruby](#)

1..1, sin restricciones de orden.

El identificador de la sección permite referir la función generada en las [plantillas de descripción fija](#), por medio de la directriz [textBPV](#).

textBPV

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Inserta en una descripción, o establece como valor en una plantilla de fragmento de descripción, el resultado de una función de valor de base por la presentación de texto .	Atributo de DFTemplateConfig y DescAreaTemplate .

Aparece en	Cardinalidad y orden
UniqueDF	0..1, sin restricciones de orden.
CreateTemplate	0..1, sin restricciones de orden.
ModifyTemplate	0..1, sin restricciones de orden.
FullDesc	0..*, sin restricciones de orden.
Area	1..1, sin restricciones de orden.
ShortDesc	1..1, sin restricciones de orden.
Title	0..1, sin restricciones de orden.

El argumento debe ser el identificador de objeto de una [TextBPVFunction](#), [KRRelationRule](#) o [Descriptor](#).

Texto multilingüe

Tipo	Formato de cada línea	Resumen	Relación con el código
elemento multi-línea	"<texto>"@<i idioma>	Permite incrustar en un archivo de definiciones texto en diversos idiomas, para diversas finalidades.	Genera un LangAlt .

Aparece en	Cardinalidad y orden
AbstractDataType	1..1, y debe ser el primer elemento de la sección.
Area	0..*, sin restricciones de orden.
Class	1..1, y debe ser el primer elemento de la sección.
ComplexDataType	1..1, y debe ser el primer elemento de la sección.
CreateTemplate	0..1, y si aparece, debe ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección.
FullDesc	0..*, sin restricciones de orden.
FundamentalDataType	1..1, y debe ser el primer elemento de la sección.

linkTo	1..1, sin restricciones de orden.
ModifyTemplate	0..1, y debe ubicarse al principio de la sección.
Property	1..1, y debe ser el primer elemento de la sección.
Realm	1..1, y debe ser el primer elemento de la sección.
RepositoryArea	1..1, debe ser el primer elemento de la sección.
ShortDesc	0..*, sin restricciones de orden.
SOC	1..1, y debe ser el primer elemento de la sección.
Title	0..1, sin restricciones de orden.
UniqueDF	0..1, y si aparece, debe ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección.

En una sección determinada, siempre deben aparecer juntas las líneas que contienen el texto multilingüe. El conjunto de líneas se considera un sólo elemento. El formato de cada línea debe ser el siguiente (donde *<texto>* es el texto mismo, e *<idioma>* es el código de dos letras de un idioma determinado):

"<texto>"@<idioma>

El significado del texto multilingüe depende del contexto. En las secciones [AbstractDataType](#), [Class](#), [ComplexDataType](#), [FundamentalDataType](#), [Property](#), [Realm](#), [RepositoryArea](#) y [SOC](#), proporciona un nombre amigable del elemento generado. En cambio, en [Area](#), [FullDesc](#), [ShortDesc](#) se convierte en un [valor de base para la presentación de texto](#) y se inserta como tal en las descripciones. Por otra parte, en [linkTo](#) constituye el texto que nombra el vínculo generado, en [Title](#) genera el texto de un título, y en [CreateTemplate](#), [ModifyTemplate](#) y [UniqueDF](#) forma la etiqueta de la plantilla de fragmento de descripción.

Ejemplos:

Aquí se emplea el texto multilingüe para establecer el nombre "amigable" de una área de repositorio:

```
<RepositoryArea>
  "Shared Archival Things Area"@en
  "Área para cosas archivísticas compartidas"@es
  (... directrices o subsecciones ...)
</RepositoryArea>
```

Y aquí se usa el texto multilingüe para indicar el título de una área de una descripción:

```
<FullDesc>
  <Area main>
    <Title>
      "Full record"@en
      "Ficha completa"@es
    </Title>
    (...)
  <Area main>
    (...)
</FullDesc>
```

thisOntology

Tipo	Número de	Resumen	Relación con el
------	-----------	---------	-----------------

	argumentos		código
directriz	1	Establece el URI de la ontología generada por el archivo de definiciones.	Atributo de Ontology .

Aparece en	Cardinalidad y orden
Realm	1..1, sin restricciones de orden.

El argumento es una cadena entrecomillada con el URI de base (no abreviado) de la ontología generada. Normalmente termina con un #. (Véase también [URI de los recursos RDF](#).)

Title

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Genera el título de un Block o una Area en una descripción .	Genera un DescTitleTemplate .

Aparece en	Cardinalidad y orden
Block	0..1, sin restricciones de orden.
Area	0..1, sin restricciones de orden.

Contenidos	Cardinalidad y orden
Texto multilingüe	0..1, sin restricciones de orden.
textBPV	0..1, sin restricciones de orden.
uniqueTextBPV	0..1, sin restricciones de orden.

Esta sección sólo puede contener un elemento ([texto multilingüe](#), [textBPV](#) o [uniqueTextBPV](#)). Dicho elemento proporciona el texto del título.

URI de los recursos RDF

En los archivos de definiciones los URI de los recurso RDF se señalan con la [abreviación](#) de la ontología que los define y sin comillas. (La cadena completa a la que se expande dicha abreviación es establecida por [thisOntology](#) o [externalOntology](#), según el caso.)

En los [bloques de código Ruby](#) un URI sin comillas y con abreviación genera un objeto [RDFNode](#).

Ejemplo:

Aquí se especifica el URI `swv:isPartToWhole` en la etiqueta de inicio de una sección [Property](#).

```
<Property swv:isPartToWhole>
  "Is Part to Whole"@en
  "Es parte de la totalidad"@es
</Property>
```

UniqueDF

Al igual que [df](#), UniqueDF es un elemento polivalente cuyo significado cambia según dónde

aparece.

UniqueDF en las [plantillas de descripción fija](#)

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Genera una plantilla de fragmento de descripción y con base en ella inserta un fragmento de descripción en una descripción .	Genera una DFTemplateConfig y después una DescFragmentTemplate , con base la cual se inserta un DescFragment en el lugar apropiado en una Description .

Aparece en	Cardinalidad y orden
ShortDesc	0..*, sin restricciones de orden.
Area	0..*, sin restricciones de orden.
Block	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
copyOfTemplate	0..1, y si aparece, debe ser el primer elemento de la sección.
Texto multilingüe	0..1, y si aparece, debe ser ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección.
label	0..1, 0..1, y si aparece, debe ser ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección. (No pueden existir en la misma sección UniqueDF un elemento label y otro de texto multilingüe .)
imageBPV	0..1, sin restricciones de orden.
textBPV	0..1, sin restricciones de orden.
uniqueImageBPV	0..1, sin restricciones de orden.
uniqueTextBPV	0..1, sin restricciones de orden.
linkTo	0..1, sin restricciones de orden.
anchor	0..1, sin restricciones de orden.

Una sección UniqueDF permite generar y configurar una plantilla de fragmento de descripción, e insertar en la descripción un fragmento creado con base en la plantilla recién generada.

Esto resulta útil cuando uno quiere incluir en una descripción un par etiqueta-valor "único"--es decir, que no se aparezca en ninguna otra parte de la descripción, ni en una descripción de otro tipo establecida por el mismo RuleSet.

UniqueDF en en las secciones [VariableDesc](#)

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Genera una plantilla de	Genera un DFTemplateConfig y después una

		fragmento de descripción y establece la posibilidad de incluir en una descripción variable un fragmento de descripción generada por medio de dicha plantilla.	DescFragmentTemplate . Un atributo de VariableDescTemplateOpts registra esta DescFragmentTemplate como disponibles para incluirse en una VariableDescTemplate .
--	--	---	---

Aparece en	Cardinalidad y orden
VariableDesc	0..*, sin restricciones de orden.

Contenidos	Cardinalidad y orden
copyOfTemplate	0..1, y si aparece, debe ser el primer elemento de la sección.
Texto multilingüe	0..1, y si aparece, debe ser ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección.
label	0..1, 0..1, y si aparece, debe ser ubicarse después de la directriz copyOfTemplate o, si ésta no aparece, al principio de la sección. (No pueden existir en la misma sección UniqueDF un elemento label y otro de texto multilingüe .)
imageBPV	0..1, sin restricciones de orden.
textBPV	0..1, sin restricciones de orden.
uniqueImageBPV	0..1, sin restricciones de orden.
uniqueTextBPV	0..1, sin restricciones de orden.

En este contexto una sección UniqueDF permite generar y configurar una plantilla de fragmento de descripción, y establecer la posibilidad de insertar en la descripción un fragmento creado con base en la plantilla generada.

Esto resulta útil cuando uno quiere establecer la posibilidad de incluir en una descripción un par etiqueta-valor "único"--es decir, que no se aparezca en ninguna descripción de otro tipo establecida por el mismo RuleSet.

uniqueImageBPV

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Define una función de valor de base para la presentación de imagen , cuyo resultado puede emplearse en plantillas de diferentes tipos.	Genera un ImageBPVFunction .

Aparece en	Cardinalidad y orden
Area	0..*, sin restricciones de orden.
FullDesc	0..*, sin restricciones de orden.
ShortDesc	0..*, sin restricciones de orden.
CreateTemplate	0..1, sin restricciones de orden.
ModifyTemplate	0..1, sin restricciones de orden.

UniqueDF	0..1, sin restricciones de orden.
--------------------------	-----------------------------------

La función que genera esta directriz es "única", es decir, no puede reutilizarse en ninguna otra configuración o plantilla. El argumento es un [bloque de código Ruby](#).

Si la directriz aparece en una [Area](#), [FullDesc](#) o [ShortDesc](#), se inserta el resultado de la función en la descripción. Si la directriz aparece en una [CreateTemplate](#), [ModifyTemplate](#) o [UniqueDF](#), el resultado de la función constituye el valor del par etiqueta-valor de la plantilla de fragmento de descripción.

uniqueTextBPV

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Define una función de valor de base para la presentación de texto , cuyo resultado puede emplearse en plantillas de diferentes tipos.	Genera un TextBPVFunction .

Aparece en	Cardinalidad y orden
Area	0..*, sin restricciones de orden.
FullDesc	0..*, sin restricciones de orden.
ShortDesc	0..*, sin restricciones de orden.
CreateTemplate	0..1, sin restricciones de orden.
ModifyTemplate	0..1, sin restricciones de orden.
UniqueDF	0..1, sin restricciones de orden.

La función que genera esta directriz es "única", es decir, no puede reutilizarse en ninguna otra configuración o plantilla. El argumento es un [bloque de código Ruby](#).

Si la directriz aparece en una [Area](#), [FullDesc](#) o [ShortDesc](#), se inserta el resultado de la función en la descripción. Si la directriz aparece en una [CreateTemplate](#), [ModifyTemplate](#) o [UniqueDF](#), el resultado de la función constituye el valor del par etiqueta-valor de la plantilla de fragmento de descripción.

useDefinition

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1	Reutiliza de manera íntegra las configuraciones establecidas por otra sección Descriptor .	Atributo de Descriptor .

Aparece en	Cardinalidad y orden
Descriptor	0..1, sin restricciones de orden.

El argumento refiere la sección [Descriptor](#) que se pretende retomar, la cual puede ubicarse en otro [RuleSet](#) o incluso otro archivo de definiciones.

Incluir esta directriz en una sección [Descriptor](#) equivale a repetir en dicha sección las mismas directrices que aparecen en la sección [Descriptor](#) referida. (Si aparecen otras directrices junto con [useDefinition](#) en la primera sección, éstas reemplazan las directrices retomadas de la segunda

sección.)

VariableDesc

Este elemento puede ser sección o directriz.

Como sección:

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Establece opciones para la generación de descripciones variables .	Genera un objeto VariableDescTemplateOpts .
Contenidos	Cardinalidad y orden		
df	0..*, sin restricciones de orden.		
UniqueDF	0..*, sin restricciones de orden.		

Como directriz:

Tipo	Número de argumentos	Resumen	Relación con el código
directriz	1; debe ser la cadena default	Señala que la configuración de la descripción variable de los objeto descriptibles vinculados con un RuleSet debe elaborarse de manera predeterminada.	Se genera automáticamente un VariableDescTemplateOpts por medio de un algoritmo predeterminado.

Las reglas acerca de dónde puede aparecer este elemento no cambian si es sección o directriz:

Aparece en	Cardinalidad y orden
Descriptions	1..1. Debe aparecer al principio de la sección, o después de NonDefaultDescFragments , si aparece éste.

Vocabulary

Tipo	Identificador de sección	Resumen	Relación con el código
sección	ninguno	Define los elementos que se incluyen en la ontología que genera el archivo de definiciones.	Genera un objeto Vocabulary .
Aparece en	Cardinalidad y orden		
DefinitionSet	0..*, sin restricciones de orden.		
Contenidos	Cardinalidad y orden		
Class	0..*, sin restricciones de orden.		
Property	0..*, sin restricciones de orden.		
AbstractDataType	0..*, sin restricciones de orden.		

ComplexDataType	0..*, sin restricciones de orden.
FundamentalDataType	0..*, sin restricciones de orden.

Notas

1. Esto es porque establecen definiciones que son esenciales para la operación del sistema, si bien ambos--especialmente `shared-worldview.def`--también ofrecen al administrador del mismo un sinfín de posibilidades para modificar y expandir su funcionamiento. No obstante, sería sencillo adaptar el sistema para no incluir `shared-worldview.def` en el caso de una instalación que no requiera de las funciones referidas allí. Nótese que la gran mayoría de los recursos que se refieren de manera hardcodeada en el sistema se establecen en `system.def`.
2. Existe el problema de la imposibilidad en RDF de asignar simultáneamente a un literal un tipo de datos y un idioma. En la capa de lógica de RC del Pescador se elimina esta restricción y se establece internamente un *workaround* para almacenar tanto la información del tipo de datos como la del idioma sin romper con las especificaciones.
3. Véase las normas [Concepts and Abstract Syntax](#) y [Fragment Interchange](#)